Narzędzia sieciowe

Copyright (C) by Wojciech Myszka
myszka@immt.pwr.wroc.pl

14 listopada 1997 roku

Spis treści

Część I

Podręcznik użytkownika

Rozdział 1

Wstęp

Tekst znajdujący się poniżej stanowi próbę zebrania w jednym miejscu informacji o narzędziach sieciowych dostępnych w sieci Internet na komputerach UNIXowych i PC–tach (**wyłącznie w środowisku DOS**).

Wersja UNIXowa opracowana jest na podstawie moich doświadczeń z wykorzystania systemu HP–UX (korygowanych wiadomościami o systemie SCO ODT w dosyć starej wersji) (przydały by się korekty uwzględniające i inne wersje zwłaszcza SunOS i Solaris oraz Linux); wersja PC obejmuje oprogramowanie PC/TCP firmy ftp Software [5] (warto by ją wzbogacić o oprogramowanie PD).

Książka powstaje na potrzeby pracowników i studentów Zakładu Dynamiki Instytutu Materiałoznawstwa i Mechaniki Technicznej Politechniki Wrocławskiej. Może jednak być przedmiotem zainteresowania i innych użytkowników Polskiego Internetu.

Od tego wydania podręcznik został rozbudowany o najbardziej podstawowe informacje o oprogramowaniu seciowym Windows 95 (polecenia **ping**, **tracert**, **telnet** i **ftp**.) Być może w jakimś czasie dostępne będą również informacje na temat konfigurowania tego środowiska do pracy w sieci z wykorzystaniem modemu.

Po bardzo długiej przerwie wróciłem do podręcznika. Z nowości – pojawiły się bardzo elementarne informacje na temat modemów i komunikacji szeregowej.

Niestety, okazało się też, że bardzo wiele odsyłączy sie całkowicie zdezaktualizowało. Nie jestem w stanie sprawdzić ich wszystkich, **proszę więc wszystkich łaskawych czytalników o informacje o błedach**.

Ma to być swoisty „podręcznik użytkownika”. Być może kiedyś przyjdzie i kolej na coś w rodzaju „podręcznika administratora”. (Prawdę mówiąc napisałem już jeden rozdział – poświęcony konfiguracji nameserwera – **named**. Pojawiły się też pewne informacje o SNMP.)

Obecna wersja jest **ciągle** w trakcie pisania. Będzie uzupełniana w miarę upływu czasu. Jej spis treści zawiera podstawowe informacje o zamiarach autora. Zapewne zamiary te będą się zmieniać z czasem. **Zwracam uwagę, że zmienia się ona w bardzo różnym tempie – bywają zastoje kilkumiesięczne a czasami kilkutygodniowe – ostatnio dopisuję nowe informacje baaaardzo rzadko.**

Udostępniam ją jednak już teraz w postaci hipertekstowej[1] (oraz zbioru PostScriptowego – długi!! ~1MB[2][3]

Od dłuższego już czasu podręcznik prezentowany jest w postaci ze wszystkimi znakami diakrytycznymi (**po polsku**). Biorąc pod uwagę upowszechnianie się programów pozwalających na poprawne rozpoznawanie polskicjch liter zrezygnowałem z możliwości przekodowywania dokumentu do innych standardów.

Informacje o tym czemu (i jak) korzystać w Internecie z polskich liter zawarte są na Polskiej Stronie Ogonkowej[4].

Ponieważ od Narzędzi sieciowych został „odłączony” podręcznik programu **pine**[5] osobno dostępna jest i jego wersja PostScriptowa.

W ostatnim czasie zmalała liczba uwag krytycznych (a szkoda), pojawiają się za to prośby o dostarczenie podręcznika w postaci jednego zwartego pliku. Ulegając tym żądaniom (i żeby ułatwić sobie życie) przygotowałem osobny, skompresowany plik zawierający całość (choć bez ilustracji, na razie). Dostępny jest on via http.

Miłej lektury i nie zapominajcie o uwagach.

Mam nadzieję, że obecne wersje obu dokumentów:

- mogą być przydatne nawet niekompletne *in statu nascendi*,

[1] http://www.immt.pwr.wroc.pl/export_hp/tool/tool.html

[2] http://www.immt.pwr.wroc.pl/export_hp/toolex/tool.ps

[3] Dostępna jest też również wersja skompresowana gdyby ktoś chciał mieć wersję „drukowaną”; ten będzie jednak aktualizowany znacznie rzadziej – jest bardzo wielki gdyż podręcznik przekroczył już 100 stron i wątpię, że ktoś będzie chciał drukować go codzriennie :-)

[4] http://www.agh.edu.pl/ogonki/

[5] http://www.immt.pwr.wroc.pl/export_hp/toolex/pine.html

- uzyskam życzliwe (ale nie tylko :–) uwagi czytelników pozwalające pracę udoskonalić (wszelkie sugestie na temat treści, zauważonych błędów, itp będą mile widziane),
- znajdzie się ktoś, kto zechce brać udział w przedsięwzięciu (na zasadach współautora) i dopisze fragmenty na temat narzędzi, o których nie napisałem.

Wszelkie uwagi proszę kierować na adres `W.Myszka@immt.pwr.wroc.pl`.

Rozdział 2

Konwencja oznaczeń

Poniżej będziemy stosowali następującą konwencję zapisu:

- tekst pisany pismem maszynowym (`abcdef...`) oznacza polecenia, które użytkownik wpisuje z klawiatury;
- tekst wpisany w nawiasach kwadratowych (`[ ]`) stanowi tą część komend/poleceń, która może (ale nie muszi wystąpić) – tak zwane „parametry opcjonalne”;
- tekst pisany italikami (*abcdef...*) stanowi część komendy, którą powinien dostarczyć użytkownik;

Rozdział 3

Sposób realizacji usług sieciowych

W dalszej części tego opracowania będziemy mówili o „usługach sieciowych”. Wśród części czytelników może pojawić się pytanie: Co to jest owa usługa sieciowa i jak jest ona realizowana?

Wiele osób uruchamiających program **telnet** (opisany na str. 6.1), **ftp** (str. 7), lynx, **Mosaic**, a nawet **ping** (str. 4.3) czy korzystających z poczty elektronicznej (str. 9) korzysta z **usług sieciowych**!

O usługach sieciowych można pisać wiele, jedną z lepszych książek w całości poświęconych temu tematowi jest praca Eda Krola [11]. Można również polecić książeczkę Brendana Kechoe (Zen and the Art of the Internet.) [9] (istnieje gdzieś również jej wersja powszechnie dostępna w sieci – trzeba poszukać za momocą archiego.)

3.1 Serwer i klient

Usługa ta, w przypadku programu **ping** jest bardzo prosta – polega na tym, że oddalony komputer odsyła otrzymany pakiet informacji. W innych przypadkach sprawa jest nieco bardziej skomplikowana, ale w każdym przypadku wymaga współdziałania dwu komputerów (czy bardziej precyzyjnie mówiąc dwu programów). Jeden z nich, incjujący połączenie nazywa sie klientem, drugi – przyjmujący je: serwerem.

Wysyłając list (*e–mail*) do kolegi, nie mamy do czynienia z przypadkiem współpracy dwu równorzędnych komputerów/programów (mimo, że za 5 minut kolega po przeczytaniu listu na niego odpowiada); mieliśmy do czynienia z dwoma sesjami połączeń klient – serwer. Ten komputer, który w jednej sesji był klientem (nadawał pocztę), podczas drugiej sesji był serwerem (odbierał odpowiedź).

Tak więc funkcja *serwera* czy *klienta* nie jest czymś związanym na stałe z jakimś komputerem – jest czymś bardzo dynamicznym, zmiennym w czasie i nieustalonym. Zwłaszcza systemy wieloprogramowe mogą pełnić wobec siebie (czy nawet pełnią) funkcje serwera i klienta równocześnie.

Pragnę również zwrócić uwagę na łatwość generowania usług korzystających z rozproszonych zasobów jaką daje sieć komputerowa (i nowoczesna infrastruktura techniczna) – potrzebne są tylko zasoby!.

3.2 „Porty”

Klient kontaktując się z serwerem musi zapukać w odpowiedni sposób do odpowiednich drzwi. Takie drzwi nazywają się „portem”. Nazwa ta została przyjęta przez analogię do rzeczywistych gniazd komputera, do których można podłączyć rzeczywiste urządzenia.

Oprogramowanie „tworzy” na potrzeby komunikacji z innymi komputerami (realizacji usług sieciowych) duże ilości portów. Każdy port ma swój numer. Te, które służą do przyjmowania połączen (obsługują serwery) muszą mieć numery, które będą znane wszystkim klientom.

Jeżeli ktoś z państwa pragnie zapoznać się z numerami portów zarezerwowanymi dla różnych serwisów (ale nie tylko) polecam dokument z serii RFC[1] o numerze 1700[2] (bardzo dlugie!) pod tytułem „Assigned Numbers”. Dokument ten definiuje bardzo wiele innych rzeczy i aż dziw, że nie wszystkie te definicje działają...

Porty o numerach niższych od 256 zostały zarezerwowane dla „dobrze znanych usług”[3] (telnet, ftp, gopher, www...); numery od 256 do 1024 zostały kiedyś zarezerwowane dla „usług specyficznych dla systemu UNIX”, (rlogin, rcp, talk...) – ale już od dawna nie jest to prawdą – wiele innych systemów operacyjnych potrafi je realizować.

[1] Request For Comments – dokumenty będące swoistymi standardami Internetu

[2] ftp://ftp.pwr.wroc.pl/pub/doc/rfc/rfc1700.txt.gz

[3] *„well known services”*

Na komputerach UNIXowych (i tylko tam zadziałają te URLe) plik `/etc/services`[4] zawiera przyporządkowanie usług numerom portów. Dodatkowy plik `/etc/inetd.conf`[5] wiąże te usługi z programami je realizującymi.

Pozwalam sobie zamieścić (niepełną) listę portów do których można „zapukać" telnetem. Nie zalecam nadużywania tej czynności. Przypominam, że połączenie telnetowe można przerwać w każdej chwili naciskając (w systemie UNIX) klawisze Ctrl–], a następnie q i Enter (czytaj również o sekwencji „escape" (str. 6.1.2):

time informacja, w postaci czytelnej, o aktualnym czasie port 13 – połączenie jest przerywane automatycznie – natychmiast po wydrukowaniu daty,

ftp port 21 – można próbować komend: `help` i `quit`, która kończy połączenie.

telnet tam sie zwykle łączymy programem **telnet**: port 23 – żeby przerwać połączenie musimy podać właściwą nazwę użytkownika i hasło, a następnie rozłączyć się z systemem, albo użyć metody opisanej wcześniej (sekwencji „escape" (str. 6.1.2),

smtp no cóż, może nie powinienem... przez port 25 odbywa się przekazywanie poczty; można wydać komendę `help`, żeby zobaczyć jakie komendy są realizowane i `quit` żeby przerwać połączenie.

gopher po uzyskaniu połączenia naciskamy Enter – powinniśmy uzyskć główne menu (w postaci niesformatowanej!): port 70 – po wypisaniu informacji połączenie jest przerywane automatycznie,

finger usługa, o której będziemy jeszcze pisać (str. 4.1), po uzyskaniu połączenia trzeba wpisać nazwę użytkownika, o którego pytamy:[6] port 79 – połączenie jest przerywane automatycznie po udzieleniu informacji,

www port 80 po nawiązaniu połączenia wydajemy następujące polecenie:

```
GET / HTTP/1.0 User-Agent:
```

i naciskamy **dwukrotnie** klawisz Enter. W odpowiedzi powinniśmy uzyskać (niesformatowaną) zawartość strony głównej serwera.[7]

nntp (network news transfer protocol) port 119 komenda `help` informuje o aktualnych możliwościach, `quit` kończy połączenie (Uwaga: podany URL zadziała tylko dla komputerów z domen *.wroc.pl, *.opole.pl, *.zgora.pl)

Zwracam również uwagę, że „dobrze znane usługi" mogą być również serwowane przez inne numery portów, ale wówczas trzeba ten numer portu stale pamiętać i podawać explicite.

[4]file://localhost/etc/services
[5]file://localhost/etc/inetd.conf
[6]URL zadziała tylko na komputerze realizującym tą usługę!
[7]Serdecznie dziękują za podpowiedź Piotrowi Gronkowi ¡gronek@novell.ftj.agh.edu.pl¿[8] z AGH w Krakowie.

Rozdział 4

Informacje o sieci

Oprogramowanie sieciowe pracujące w każdym środowisku, dostarczane przez różnych dostawców lub PD (Public Domain) zawiera pewną ilość narzędzi pozwalających zdobywać podstawowe informacje o sieci, jej użytkownikach czy drodze przesyłania informacji między komputerami.

Niestety programy te bardzo różnią się między sobą (głównie zestawem, ale i sposobem uruchamiania). W systemie UNIXowym można zawsze liczyć na **finger** (str. 4.1) i **ping** (str. 4.3).

Inne wymienione poniżej narzędzia mogą być (ale nie muszą) na komputerze, z którego korzystacie. Wiele z nich można zdobyć w sieci i skompilować samemu. Odpowiednikiem programu **host** dostępnego w pakiecie PC/TCP jest program **nslookup** (str. 5.4).

Oprogramowanie PC/TCP firmy FTP Software, Inc. pozwala uzyskiwać podstawowe informacje o sieci, komputerach do niej podłączonych i ich użytkownikach. Służą do tego programy: **host** (str. 5.5), **ping** (str. 4.3), **finger** (str. 4.1), **nicname** (str. 4.2), **setclock** (str. 4.5), **whois** (str. 4.6).

4.1 finger

Program pozwala uzyskać podstawowe informacje o zarejestrowanych użytkownikach systemu komputerowego. Nie wszystkie komputery udzielają takich informacji!

Normalnie drukowane są następujące informacje:

- nazwa użytkownika,
- jego imie i nazwisko,
- czas podłączenia do systemu oraz czas nieaktywności klawiatury,
- nazwa kartoteki głównej użytkownika („*home directory*")
- zawartość plików `.plan` i `.project`,
- numer telefonu biurowego (jeżeli jest znany).

Komenda **finger** pozwala z jednej strony zdobywać podstawowe informacje o użytkowniku oddalonego komputera. Z drugiej zaś, pozwala każdemu użytkownikowi na budowanie własnego obrazu na zewnątrz.

Typowym użyciem jest wpisywanie do zbioru `.plan` informacji o planowanym powrocie z dłuższej nieobecności.

Oto przykład informacji uzyskanej komendą **finger**:

```
$ finger myszka@ldhpux
[ldhpux.immt.pwr.wroc.pl]
Login name: myszka                      In real life: Wojciech Myszka
Bldg: 113 B1, Work phone: 2754  51-48-58
Directory: /users/myszka                Shell: /bin/ksh
On since May 12 13:34:04 on console              9 hours 54 minutes Idle Time
No Plan.
```

Niektórzy użytkownicy używają protokołu **finger** do udzielania prostych informacji: polecamy informacje o największych centrach suprkomputerowych[1], trzęsieniach ziemi[2], informacje o wydarzeniach z historii[3],

[1] http://www.pwr.wroc.pl/cgi-bin/finger?gunter@yarrow.wt.uwa.edu.au
[2] http://www.pwr.wroc.pl/cgi-bin/finger?quake@geophys.washington.edu
[3] http://www.pwr.wroc.pl/cgi-bin/finger?copi@oddjob.uchicago.edu

usługach sieciowych Scotta Yanoffa[4]). (Jeżeli już wspominamy o nim, polecam jego listę usług sieciowych[5] – zaniedbaną już, niestety.)

Komenda **finger** i informacje, których ona dostarcza zostały uznane za „potencjalnie" niebezpieczne i dlatego niektóre komputery albo wyłączają ją zupełnie, albo ograniczają jej dostępność – tylko dla niektórych komputerów.

4.1.1 Wersja UNIX

`finger [`*`options`*`]` *`user--name`* `...`
Dostępne opcje (*options*):

- `-m` przeglądaj wyłącznie nazwy użytkowników,
- `-l` wydruki w postaci „długiej",
- `-p` nie drukuj zawartości pliku `.plan`,
- `-s` „krótka" postać wydruku,
- `-R` drukuj nazwę komputera.

user–name nazwa użytkownika w postaci: *user* lub *user@host*, albo też *@host*; w pierwszym przypadku uzyskamy informacje o użytkowniku lokalnego komputera, w drugim o użytkowniku (–ach) komputera odległego; w trzecim o wszystkich, aktualnie pracujących użytkownikach.[6] Co więcej jako *user–name* można podać imię lub nazwisko użytkownika albo oficjalną jego nazwę na komputerze, tzn. zapytanie[7]:

```
finger piotr@ldscoux.immt.pwr.wroc.pl
```

udzieli informacji o wszystkich Piotrach, będących użytkownikami komputera ldscoux.

4.1.2 Wersja PC/TCP

Użycie:

`finger [`*user*`]` *@host*
`finger [-?|-version]`

user określa nazwę użytkownika, jeżeli zostanie pominięta, niektóre komputery odpowiadają podając listę aktualnie pracujących użytkowników. **finger** oczekuje na odpowiedź ok. 20 sekund,

host określa nazwę komputera lub jego internetowy numer (i jest parametrem obowiązkowym... w zestawie oprogramowania nie ma serwera finger),

`-?` wyświetla krótkie objaśnienie i sposób użycia programu,

`-version` podaje numer wersji oprogramowania.

4.2 nicname

Program (z zestawu PC/TCP) pozwala uzyskać informacje o (zarejestrowanych) użytkownikach sieci Internet. Niestety, nie wszyscy użytkownicy rejestrują się w centrum informacji (nic.ddn.mil).[8] Dotyczy to zwłaszcza użytkowników z krajów podłączających się do sieci. Stąd zastosowanie tego programu jest niewielkie.

Użycie:

`nicname` *`user`*
`nicname -help`
`nicname -?|-version`

Znaczenie parametrów jest następujące:

---

[4] http://www.pwr.wroc.pl/cgi-bin/finger?yanoff@csd4.csd.uwm.edu

[5] http://www.spectracom.com/islist/

[6] Zwracam uwagę, że mimo iż argument komendy **finger** jest bardzo podobny do adresu e–mailowego nie zawsze można jej użyć z adresem jako argumentem. O przyczynach tego faktu mówimy nieco na stronie 5.1

[7] http://www.pwr.wroc.pl/cgi-bin/finger?piotr@ldscoux.immt.pwr.wroc.pl

[8] Szymon Sokół pisze: „już od jakiegoś czasu nic.ddn.mil nie rejestruje w swojej bazie ludzi spoza MILNETu – administratorzy domen i sieci są rejestrowani obecnie w rs.internic.net; inni ("szarzy") użytkownicy w ds.internic.net"

user nazwa użytkownika, o którego pytamy; może to być nazwa użytkownika, nazwa komputera lub nazwisko zarejestrowanego użytkownika,

`-help` wyświetla pożyteczne informacje na temat wykorzystania programu; jeżeli w linii komendy umieścimy tylko słowo `help` komenda zostanie przekazana do serwera nic.ddn.mil, który przekazuje informacje o sposobach bardziej zaawansowanego poszukiwania adresów zarejestrowanych użytkowników.

4.3 ping

Program ten wysyła pakiet informacji żądający odesłania go do wysyłającego. Pozwala to badać istnienie połączenia między komputerami, drogę między nimi, czas potrzebny na przejście pakietu, sprawdzać czy drugi z komputerów pracuje. Nie należy nadużywać tej usługi, gdyż ogranicza ona przepustowość sieci.

4.3.1 Wersja Windows 95

Rozdział ten jest jednym z kilku (inne opisują programy **ftp** (str. 7.7), **tracert** (str. 4.4.1) i **telnet** (str. 6.1.1)) opisujących podstawowe usługi sieciowe dostępne w systemie Windows 95.

Sposób uruchomienia (Program **ping** w wersji Windows 95 pracuje w oknie DOS): `ping [-t] [-a]`
`[-n count] [-l length] [-f] [-i ttl]`
`[-v tos] [-r count] [-s count] [[-j host-list] | [-k host-list]]`
`[-w timeout] destination-list`

Znaczenie poszczególnych parametrów jest następujące:

`-t` „pinguje” wskazany komputer, aż działanie programu zostanie przerwane,

`-a` zabrania zamieniać adres podany w postaci numerycznej do postaci symbolicznej,

`-n` ***count*** określa liczbę wysyłanych pakietów; standardowo 4,

`-l` ***length*** określa długość wysyłanego pakietu (wielkość standardowa wynosi 64 bajty, maksymalna xxx bajtów),

`-f` powoduje ustawienie w wysyłanych pakietach znacznika zabraniającego ich fragmentowania,

`-i` ***ttl*** określa maksymalny czas życia pakietu,

`-v` ***tos*** określa rodzaj usługi (**Type Of Servicve**); pozwala badać w jaki sposób różne routery traktują różne usługi (jeżeli mają zaimplementowaną tę funkcję),

`-r` ***count*** pozwala śledzić drogę pakietu (rezerwuję pola na zapamiętanie adresów maksimum *count* routerów przez które przechodzi pakiet), 1 <= *count* <= 9,

`-s` ***count*** pozwala zapisać w pakiecie (rezerwuje miejsce na max *count* znaczników) czas, w którym pakiety przeszły przez kolejne routery,

`-j` ***host-list*** określa adresy routerów (drogę) przez które pakiet może przejść; można wskazać max 9 adresów,

`-k` ***host-list*** określa adresy routerów (drogę) przez które pakiet **nie** może przejść; można wskazać max 9 adresów,

`-w` ***timeout*** określa czas (w milisekundach) jaki program oczekuje na powrót pakietu,

destination-list pkreśla adres „pingowanego” komputera.

4.3.2 Wersja PC/TCP

Użycie:

```
ping [-optons] [host]
ping -?|-version
```

Program pozwala na użycie bardzo wielu opcji pozwalających badać różne techniczne parametry połączenia między komputerami. Tu omówimy jedynie niektóre z nich:

`-l length` ustala długość pakietu testowego na *length* bajtów; standardowa długość pakietu – 256 bajtów; maksymalna długość – 1472 bajty,

-n *times* określa liczbę powtórzeń pakietu; standardowo wysyłany jest tylko jeden pakiet,

-q włącza opcję śledzenia drogi (*Trace Route*); pozwala ona rozpoznać wszystkie routery po drodze między komputerami,

-t powtarza operacje „w nieskończoność” wysyłając kolejny pakiet po otrzymaniu potwierdzenia o dojściu poprzedniego, **ping** wyświetla wszystkie informacje o błędach, oraz informację o liczbie wysłanych i otrzymanych pakietów; przerwanie pracy programu następuje po naciśnięciu klawisza „q”,

-w *seconds* określa jak długo **ping** czeka na „powrót” wysłanego pakietu (standardowo 5 sekund) zanim uzna, że wysłany pakiet „zaginął”,

-z włącza tryb, w którym **ping** informuje tylko o fakcie dotarcia (lub nie) pakietu, pomijając wszystkie informacje dodatkowe,

host określa nazwę lub adres internetowy komputera, z którym się komunikujemy,

-? wyświetla, krótką informacje o użyciu programu,

-version wyświetla numer wersji programu.

O ile wiem, nowa wersja pakietu PC/TCP dodała do programu **ping** kilka nowych opcji...

4.3.3 Wersja UNIX

Dosyć trudno opisywać komendę **ping** na komputerach UNIXowych. Z tego co wywnioskowałem przeglądając strony manuala kilku systemów UNIX każdy z nich realizuje komendę nieco inaczej. Być może wynika to z faktu, że zosyała ona wymyślona i opracowana jako program *Public Domain* i zaimplementowana na różny sposób w różnych wersjach UNIXa.

Najprostsze użycie:

ping *host* [*packetsize*] [*count*]
gdzie:

host nazwa (lub numer IP) oddalonego komputera,

packetsize długość pakietu w bajtach, minimalna długość – 8 bajtów, domniemana 64 bajty, maksymalna 4096 bajtów, gdy długość pakietu jest krótsza niż 16 bajtów nie ma w nim miejsca na umieszczenie informacji związanych z czasem – nie będzie można określić czasu podróży pakietu („*round–trip*”),

count liczba pakietów, które ma wysłać **ping**; z domniemania wysyła je tak długo, aż zostanie przerwany.

W niektórych systemach, jeżeli **ping** zostanie wywołany jedynie z parametrem *host* odpowiada: „host is alive” gdy uzyskał odpowiedź lub sygnalizuje przekroczenie czasu (*time–out*) gdy odpowiedzi nie uzyskał. W innych realizacjach wypisuje numer pakietu i czas podróży, a po przerwaniu podaje statystykę: liczbę pakietów wysłanych, liczbę, która powróciła, minimalny, średni i maksymalny czas podróży. Pakiety, które nie wróciły nie są zaznaczane w żaden sposób.

Z dodatkowych opcji, które realizuje **ping** w systemie HP–UX można polecić [-o]. Gdy zostanie ona użyta program umieszcze w wysyłanych pakietach żadanie, aby każdy z routerów, przez który pakiet przechodzi dopisywał w nim swój adres. Pozwala to śledzić drogę pakietu (ale tylko na krótkich dystansach). Zwracam uwagę, że nie wszystkie routery realizują taką możliwość.

Niektóre implementacje programu **ping** mogą wysyłać następny pakiet niezwłocznie po powrocie poprzedniego. W szczególnych warunkach może to prowadzić do wysycenia przepustowości sieci.

4.4 traceroute

Funkcje zbliżone do tych realizowanych przez program **ping** (w środowisku DOS i PC/TCP) realizuje program traceroute. Idea jego działania jest następująca: z każdym pakietem wysyłanym „w świat” związany jest pewien licznik zawierający „czas życia” (*time to live*). Licznik ten jest zmniejszany o jeden podczas przejścia przez każdy router. Gdy osiągnie on zero, pakiet jest zwracany do nadawcy jako „przeterminowany”.

traceroute wysyła pakiet do wskazanego adresu z czasem życia 1 (po osiągnięciu pierwszego routera jest on zwracany), następnie z czasem życia 2, i tak dalej... Pozwala to prześledzić całą droge od nadawcy do odbiorcy. (Program realizuje funkcję podobną jak program **ping** (str. 4.3.2) z pakietu PC/TCP wywołany z opcją -q.)

Problem stwarzają tylko te routery, które nie sygnalizują przeterminowania licznika, lub nie zmniejszają go.

4.4.1 System Windows 95

Rozdział ten jest jednym z kilku (inne opisują programy **ftp** (str. 7.7), **ping** (str. 4.3.1) i **telnet** (str. 6.1.1)) opisujących podstawowe usługi sieciowe dostępne w systemie Windows 95.

Sposób uruchomienia: `tracert [-d] [-h` *`maximum-hops`*`] [-j` *`host-list`*`] [-w` *`timeout`*`]` *`target-name`*

Znaczenie parametrów jest następujące:

`-d` poleca nie zamieniać adresów numerycznych na symboliczne,

`-h` ***maximum-hops*** określa maksymalną liczbę punktów pośrednich (skoków) jaką może wykonać program,

`-j` ***host-list*** patrz odpowiedni parametr programu **ping** na str. 4.3.1,

`-w` ***timeout*** czas (w milisekundach) jaki program oczekuje na powrót pakietu, zanim uzna, że zaginął,

target-name adres komputera.

4.4.2 System UNIX

Sposób wykorzystania programu (podajemy tylko niektóre parametry):

`traceroute [-n] [-g` *`host1`*`]` *`host`*
gdzie:

`-n` używamy, gdy chcemy aby adresy były podawane w postaci numerycznej, a nie symbolicznej – przyśpiesza to działanie programu,

`-g` *`host1`* adres węzła *pośredniego*,

host adres (lub numer IP) komputera, do którego śledzimy drogę.

Przykład wykorzystania programu **traceroute**:

```
$ traceroute ftp.uu.net
traceroute to ftp.uu.net (192.48.96.9), 30 hops max, 40 byte packets
 1  wazniak8.pwr.wroc.pl (156.17.8.30)  2 ms  2 ms  1 ms
 2  wroc-gwb.pwr.wroc.pl (156.17.254.33)  3 ms  3 ms  2 ms
 3  waw-wroc.nask.waw.pl (193.59.200.41)  72 ms  280 ms  226 ms
 4  193.59.201.10 (193.59.201.10)  352 ms  81 ms  158 ms
 5  192.121.154.81 (192.121.154.81)  994 ms *  1394 ms
 6  icm-dc-1.icp.net (192.121.154.234)  1809 ms  1433 ms  937 ms
 7  192.157.65.227 (192.157.65.227)  1421 ms  1240 ms  997 ms
 8  Falls-Church4.VA.ALTER.NET (137.39.43.97)  974 ms  907 ms  1108 ms
 9  IBMpc01.UU.NET (137.39.8.20)  1146 ms  1528 ms  4960 ms
10  ftp.UU.NET (192.48.96.9)  1299 ms  3333 ms  2581 ms
```

Program można wykorzystać również do wyznaczenia trasy pomiędzy trzema węzłami:

1. komputerem, na którym wydano komendę **traceroute**,
2. węzłem pośrednim – *host1*,
3. węzłem docelowym – *host*.

Może to się przydać do sprawdzenia czy pakiety pomiędzy naszym komputerem a wezłem docelowym pokonują trasę tą samą drogą co z powrotem:

```
traceroute -g sun1000.ci.pwr.wroc.pl ldhpux.immt.pwr.wroc.pl
```

a oto wynik:

```
traceroute to ldhpux.immt.pwr.wroc.pl (156.17.8.1) 30 hops max, 38 byte packets
 1  wazniak.wask.wroc.pl (156.17.8.30)  4 ms  1 ms  1 ms
 2  centrum.wask.wroc.pl (156.17.254.251)  3 ms  3 ms  3 ms
 3  sun1000.ci.pwr.wroc.pl (156.17.5.3)  3 ms *  4 ms
 4  centrum.wask.wroc.pl (156.17.5.30)  6 ms  5 ms  5 ms
 5  wazniak.wask.wroc.pl (156.17.254.254)  5 ms  4 ms  4 ms
 6  ldhpux.immt.pwr.wroc.pl (156.17.8.1)  5 ms  5 ms  4 ms
```

albo zdalnego sprawdzenia czy z komputera *host1* jest połączenie z komputerem *host*:

```
traceroute -g apollo.int.pan.wroc.pl plearn.edu.pl
```

Program **traceroute** nie jest na ogół „standardowym” elementem oprogramowania sieciowego.

4.5 setclock

Program pozwala ustawić zegar PC na podstawie zegara komputera, który uznajemy za wzorzec czasu. W bardzo wielu sytuacjach sieciowych jest bardzo ważne aby zegary komputerów były zsynchronizowane.

Użycie:

```
setclock [time_server]
setclock -?|-version
```

time_server nazwa lub adres internetowy komputera, który traktujemy jako wzorzec czasu.

Komputery UNIXowe na ogół używają (jezeli w ogóle) innych narzędzi do synchronizacji swoich zegarów. Synchronizację w ramach sieci lokalnej zapewniają programy **timed** i **rdate**. W sieciach rozległych używany jest protokół synchronizacji czasu **NTP** lub **XNTP** – (eXtended) Network Time Protocol.

4.6 whois

Spełnia w systemie oprogramowania PC/TCP bardzo podobne funkcje jak program **finger** (str 4.1).

Użycie:

```
whois [user]@host
whois -?|-version
```

Rozdział 5

Translacja adresów symbolicznych na numeryczne i odwrotnie

Usługa tłumaczenia adresów symbolicznych na numeryczne (i odwrotnie) realizowana jest przez oprogramowanie zwane BIND (*Berkeley Internet Name Domain*). Jak wszystkie, usługa ta realizowana jest w układzie „klient–serwer".

Najpierw omówimy to z czym kontaktuje się użytkownik – stronę klienta. Program (znowu w systemie UNIXowym) nazywa sie *resolver*.

Później nieco miejsca poświęcimy nieco miejsca serwerowi – w systemie UNIXownym – nazywa się on **named** (patrz str. 12).

5.1 Adresy sieciowe

Adresy sieciowe Adres sieciowy komputera składa się z kilku członów oddzielonych kropkami. Część najbardziej z lewej strony oznacza (zazwyczaj, ale nie zawsze – np. w adresach pocztowych) nazwę komputera, część najbardziej z prawej strony – nazwę kraju.[1] Adres może wyglądać na przykład tak:

```
ldscoux.immt.pwr.wroc.pl
|       |    |   |    |_________ kraj (pl - Polska)
|       |    |   |______________ informacje terytorialne (miasto--okręg)
|       |    |__________________ informacje terytorialne (instytucja)
|       |_______________________ informacje terytorialne (jednostka org.)
|_______________________________ nazwa komputera
```

Struktura adresu jest w naturalny sposób hierarchiczna. Ułatwia to zarządzanie bazami danych adresów i poszukiwanie adresu numerycznego.

System baz danych zawierających informacje o wszystkich zarejestrowanych komputerach stanowi rzadki przykład (działającej) rozproszonej bazy danych.

Adresy numeryczne Adres numeryczny (inaczej adres IP) jest unikalną, 32 bitową liczbą, przydzielaną każdemu komputerowi pracującemu w sieci Internet. Dla ułatwienia operowania adresami IP podzielono ją na cztery bajty. Adres IP zapisuje się w postaci czterech liczb z zakresu 0–255 (odpowiadających wartości poszczególnych bajtów) oddzielanych kropkami. Wygląda on mniej, więcej tak:

```
156.17.8.1
```

Nameserver (zwany również *domain nameserver*) jest specjalnym programem udzielającym odpowiedzi wszystkim programom użytkowym na zapytania dotyczące adresów komputerów. Jeżeli potrzebnych danych nie ma w swojej pamięci – kontaktuje się z innymi nameserverami. W sieci wyróżnia się nameservery, które są pierwotnymi (primary) dla pewnych grup adresów sieciowych (domen) i te, które zawierają kopie informacji – wtórne (*secondary*). Bardzo często komputer, który jest nameserverem pierwotnym dla pewnej domeny, zawiera również kopie danych dla innej domeny. Oba wyżej wymienione rodzaje nameserwerów zwane są autorytatywnymi.

[1] Nie dotyczy to znacznej części adresów zakończonych `.com`, `.edu`, `.mil` – są to głównie adresy ze Stanów Zjednoczonych, ale niekoniecznie, również na przykład z Kanady; widziałem kiedyś adres amerykańskiej firmy komercyjnej, której moskiewska filia miała również adres zakończony na `.com` (i połączenie via Kalifornia...).

Można powoływać do życia nameserwery nie będące serwerami autorytatywnymi, zawierającymi jedynie kopie informacji.

Hierarchia adresów Domeny uszeregowane są hierarchicznie zgodnie z hierarchią adesów; w naszym przypadku wyróżniamy następujące cztery domeny:
pl
wroc.pl
pwr.wroc.pl
immt.pwr.wroc.pl

Istnieją dwie hierarchie adresów: tak zwana *prosta* (o której pisaliśmy powyżej) związana jest ona z bazą danych pozwalającą zamieniać adresy symboliczne na numery IP i *odwrotna* pozwalająca zamieniać numery IP na adresy. O ile hierarchiczne grupowanie adresów w pierwszym przypadku jest zupełnie naturalne o tyle w drugim już nie tak bardzo. Stworzono jednak na potrzeby specjalną domenę o nazwie **in-addr.arpa**, w której zapisywane są wszystkie informacje o powiązaniach numerów IP z adresami.

I tak wszystkie informacje o podsieci 156.17.xxx.yyy (przydzielonej środowisku wrocławskiemu) zawarte są w domenie 17.156.in-addr.arpa. Można odpowiedzialność za kolejne podsieci delegować dalej: 8.17.156.in-addr.arpa zawiera wszystkie informacje o podsieci 156.17.8.yyy a 1.8.17.156.in-addr.arpa jest odnośnikiem wiążącym numer 156.17.8.1 z nazwą komputera (ldhpux.immt.pwr.wroc.pl w tym przypadku).

Adresy poczty komputerowej Adres używany przez pocztę komputerową składa się z dwu części: nazwy użytkownika i nazwy komputera połączonych znakiem „@" i wygląda na przykład tak:
postmaster@ldhpux.immt.pwr.wroc.pl

Poczta może być jednak tak skonfigurowana aby „ukrywać" nazwę komputera, z którym związany jest użytkownik (ma to swój wielki sens, gdy w jednostce organizacyjnej jest więcej niż jeden komputer); pomija się wówczas jeden człon nazwy:
postmaster@immt.pwr.wroc.pl
(Zwracam uwagę iż „postmaster" jest to na każdym komputerze przyjmującym i wysyłającym pocztę komputerową osoba odpowiedzialna za jej funkcjonowanie. Można się do niej zgłaszać w razie jakichkolwiek problemów z pocztą komputerową.)

W takiej sytuacji „immt.pwr.wroc.pl" nie jest nazwą żadnego rzeczywistego komputera, a jedynie synonimem *mail exchangera*[2] czyli komputera, który na podstawie nazwy użytkownika potrafi dostarczyć pocztę pod odpowiedni fizyczny adres. W takiej sytuacji, bardzo często, administratorzy poczty elektronicznej pozwalają by nazwy użytkowników miały coś wspólnego z ich rzeczywistymi nazwiskami dopuczszając takie np. konstrukcje:
Jan_Kowalski@....

Możliwe jest takie skonfigurowanie poczty elektronicznej, aby list wysyłany na adres:
Jan_Kowalski@pwr.wroc.edu.pl trafiał do użytkownika: kowalski@ldscoux.immt.pwr.wroc.pl.

5.2 Rady praktyczne

Pamiętanie adresów różnych komputerów stanowi pewien problem dla użytkowników sieci. Z moich doświadczeń wynika, że użytkownicy (w znacznej swej części) próbują postępować na jeden z dwu sposobów:

1. używają adresów numerycznych (uważając, że znacznie łatwiej zapamiętać cztery liczby, np: 156.17.8.1 niż przydługawą (i jak dziś widzę głupią) nazwę ldhpux.immt.pwr.wroc.pl).

2. Starają się używać nazw symbolicznych.

Ze swej strony pragnę zwrócić uwagę na kilka właściwości *resolvera* (procedur odpowiadających na komputerze UNIXowym za translację adresów symbolicznych do numerycznych), które pozwalają ułatwić sobie życie (we wszystkich poniższych przykładach zakładamy, że komputer znajduje się w domenie `immt.pwr.wroc.pl`):

- jeżeli w jakiejkolwiek komendzie sieciowej (**ping**, **telnet**, **rlogin**, **ftp**...) podamy jako jej parametr nazwę `host`, np:

  ```
  ping host
  ```

  poszukując komputera, o który nam chodzi *resolver* będzie do `host` dodawał kolejne fragmenty domeny poszukując kolejno:

[2] Rekordy „MX" nameserwera mówią o powiązaniach pomiędzy adresami e–mail a komputerami, patrz również na stronie 21

```
host.immt.pwr.wroc.pl
host.pwr.wroc.pl
host.wroc.pl
```

i na końcu wreszcie:

```
host
```

(„Okrawając” domenę zostawi zawsze co najmniej jej dwa ostatnie człony.) Podobnie będzie postępował gdy podamy niepełną nazwę komputera:

```
ping graf.im
```

poszukuje komputera o nazwie `graf.im.immt.pwr.wroc.pl` (błąd!), `graf.im.pwr.wroc.pl` – sukces!, taki komputer istnieje. Podobny proces przebiega niestety nawet wtedy gdy podamy pełną nazwę komputera:

```
ping graf.im.pwr.wroc.pl
```

resolver doda wszystkie możliwe człony naszej domeny. Ponieważ za każdym razem skończy się to błędem w końcu sprawdzi nazwę `graf.im.pwr.wroc.pl` co powinno najprawdopodobniej skończyć sie sukcesem.

resolver można również tak skonfigurować, żeby wskazać explicite jakie człony, jakich domen ma dokładać do podawanych mu adresów.[3]

Proces ten może byc dosyć długotrwały, gdy *nameserver* znajduje się na drugim końcu wolnej, obciążonej linii.

Aby proces przyśpieszyć pełne nazwy komputerów[4] można/należy kończyć kropką:

```
ping graf.im.pwr.wroc.pl.
```

- Wniosek wypływający z tego jest taki: jeżeli kontaktujemy się z komputerem z „sąsiedniej” domeny (mającej znaczną część nazwy zgodną z nazwą naszej domeny) używajmy tylko tylu członów ile to konieczne aby jednoznacznie określić nazwę komputera. Dodaje to co prawda nieco pracy *nameserwer*owi ale znacznie upraszcza pisanie.
- *resolver* w wielu wersjach SO Unix pozwala na tworzenie aliasów nazw komputerów (indywidualnych dla każdego użytkownika). Warto z tej możliwości korzystać. Otóż w procesie tłumaczenia adresu symbolicznego *resolver* poszukuje zmiennej środowiska o nazwie `HOSTALIASES`. Plik wskazywany przez tą zmienną jest przeglądany w poszukiwaniu pełnej nazwy związanej z podanym aliasem. I tak, w moim przypadku zmienna `HOSTALIASES` ma wartość `/users/myszka/.hostaliases` a w pliku `.hostaliases` jest (między innymi) następująca linia:

  ```
  graf graf.im.pwr.wroc.pl.
  ```

  Pisząc `rlogin graf` uzyskuje połączenie z komputerem `graf.im.pwr.wroc.pl`
- Jeżeli chodzi o pocztę polecam korzystanie z różnych mechanizmów ułatwiających życie, a oferowanych przez programy czytania/komponowania poczty. O tym będzie mowa później.

[3] Sprawa jest ważna z kilku powodów (względy bezpieczeństwa odgrywają tu niepoślednią role). Powinno być zasadą, że żaden człon nazwy komputera nie może być zgodny z dwuliterowym kodem kraju (ostatni człon nazwy). Może to prowadzić do poważnych w skutkach nieporozumień. Zasada ta jest czasami jednak łamana, w nazwach komputerów pojawiają się człony **ch**, **ci**.

[4] Fully Qualified Domain Name – FQDN.

5.3 Konfiguracja *resolvera*

Konfiguracja resolvera jest bardzo prosta. Wystarczy w pliku /etc/resolv.conf wpisać minimum dwa wiersze informacji (każdy zaczyna się od słowa kluczowego):

nameserver po którym podajem adres IP komputera, na którym uruchominy jest program named (może być więcej niż jeden wiersz deklarujący kolejne serwery),

domain po którym podajemy nazwę domniemanej domeny w której pracuje komputer; nazwa ta będzie dodawana do wszystkich nazw komputerów, ktore nie są zakończone kropką (tak jak to opisano w rozdziale rady praktyczne (na str. 5.2)).

Różne implementacje procedur resolvera pozwalają na wpisywanie różnych dodatkowych informacji – należy zapoznać sie z odpowiednią stroną manuala (man resolv.conf).

Zwracam uwagę, że **resolver** to nie jest żaden program, a raczej zestaw (biblioteka) podprogramów zawierający funkcje pozwalające na translcję adresów symbolicznych na numerycznie i odwrotnie. Resolver może w swojej pracy (w zależności od konfiguracji) korzystać z różnych źródeł informacji:

- pliku /etc/hosts,
- systemów NIS lub NIS+ (*Network Information System*) – zwany również pod nazwą *yellow pages*,
- nameserwera,

(bardzo często w różnej kolejności i różnej konfiguracji).

5.4 System Unix

Podstawowym narzędziem służącym do sprawdzania współdziałania systemu baz danych o adresach internetowych (w systemie Unix) jest program **nslookup**. Pozwala on wydobywać informacje o pojedyńczych komputerach, ale również o całych domnach.

Pełne wykorzystanie możliwości programu **nslooup** wymaga pewnej znajomości sposobu zapisywania informacji o adresach internetowych i strukturze baz danych. Tak więc poniżej podamy podstawowe fakty o programie i informacjach, które od niego można wydobyć.

Bardzo wielu administratorów używa również programów **host** lub **dig**. Ich zaletą jest to, że oprócz zdobywania informacji mogą również dokonywać weryfikację ich porawności.

5.4.1 Translacja adresu na numer IP

Najprostsze zadanie, rozwiązujemy je pisząc:
`nslookup` *adres*
w odpowiedzi powinniśmy uzyskać numer IP (jeżeli jest związany z tym adresem).

Odpowiedź programu wygląda w sposób następujący:

```
$ nslookup garbo.uwasa.fi
Name Server:  ldhpux.immt.pwr.wroc.pl
Address:  156.17.8.1

Name:    garbo.uwasa.fi
Address:  128.214.87.1
```

Można też skorzystać z interakcyjnego, hipertekstowego formularza[5].

Dodatkowo można zadać pomocnicze pytania, piszemy o tym dalej.

5.4.2 Typ komputera

W tym przypadku trzeba przejść do interakcyjnego trybu pracy programu **nslookup**. Pytanie wygląda w następujący sposób:

[5] http://www.immt.pwr.wroc.pl/htbin/nsl

```
$ nslookup
Default Name Server:  ldhpux.immt.pwr.wroc.pl
Address:  156.17.8.1

> set type=hinfo
> garbo.uwasa.fi
Name Server:  ldhpux.immt.pwr.wroc.pl
Address:  156.17.8.1

garbo.uwasa.fi  CPU = Sun4/50   OS = SunOs
```

Zwracam uwagę, że posiadanie zbyt dokładnych informacji o architekturze komputera i wersji systemu operacyjnego może stanowić istotną wskazówkę dla „włamywacza”, który może znacznie lepiej niż administrator systemu znać słabe miejsca systemu i wykorzystać tę wiedzę w celach „niegodziwych”. Stąd, wielu administratorów nie wpisuje takich informacji do baz danych.

5.4.3 Translacja numer IP na adres

Postępowanie (czasami) jest równie proste jak w poprzednim przypadku:
`nslookup` *numer–ip*

Odpowiedź jest następująca:

```
$ nslookup 156.17.9.29
Name Server:  ldhpux.immt.pwr.wroc.pl
Address:  156.17.8.1

Name:    asic.ict.pwr.wroc.pl
Address:  156.17.9.29
```

Można też skorzystać z interakcyjnego, hipertekstowego formularza[6].

Niestety, nie wszystkie implementacje programu **nslookup** rozumieją powyższe zapytanie...

5.4.4 Pytanie o odbiorców poczty

Kilkakrotnie zwrócono się do mnie z następującym pytaniem: „*Znam adres pocztowy kolegi:* user@costam*; pragnę sprawdzić czy aktulanie pracuje na komputerze, używając programu* **finger***; wywołuję go tak:* `finger user@costam`*; dostaję dziwne odpowiedzi... Co się da z tym zrobić?*”

Mamy do czyniania z klasycznym przypadkiem ukrycia nazwy komputera (str. 5.1) w adresie pocztowym. Problem więc polega na tym jak znaleźć komputer odbierający pocztę przychodzącą na wskazany adres. Jest to pytanie o rekord MX z bazy danych nameserwera. Pytanie wygląda tak:

```
nslookup
set type=mx
costam
```

W odpowiedzi powinniśmy dostać listę (czasami jednoelementową) adresów komputerów z wraz z numerkiem określającym priorytet. Wybierzmy ten o numerku najmniejszym i tam próbujmy szczęścia z **finger**–em. Jeżeli to nie odniesie skutku – możemy zapytać komputera o najwyższym priorytecie co robi z pocztą przychodzącą do użytkownika *user*. W tym celu telnetujemy (patrz rownież str. 3.2) pod uzyskany adres i numer portu 25 (odpowiedzialnego za pocztę) i wydajemy komendę:
`vrfy` *user*
Jeżeli komputer rozumie komendę `vrfy` powinniśmy otrzymać dokładniejszy adres pocztowy i możemy procedurę powtórzyć.

5.5 MS–DOS i PC/TCP

Program **host** pozwala odnaleźć adres internetowy (numer IP) na podstawie nazwy komputera lub odwrotnie (nazwę na podstawie numeru), używając wszystkich dostępnych protokołów. Jest pełnym odpowiednikiem znanego z systemu UNIX programu **nslookup** w systemie MS–DOS.

[6] http://www.immt.pwr.wroc.pl/htbin/nsl

Użycie:

`host [-dhntACHMNPSTW*] [-a|-s` *`server`*`]` *host1 ... hostn*

`-d` próbuje wszystkie skonfigurowane nameserver'y,

`-h` pyta o rekordy klasy HESIOD,

`-n` chroni przed rekurencyjnym poszukiwaniem baz danych,

`-t` wykorzystuje tylko lokalną bazę danych (zbiór host),

`-A` pyta o rekord klasy A (patrz str. 12.2.3) (adres)

`-C` pyta o rekordy klasy CNAME (patrz str. 12.2.5) (canonical name),

`-H` pyta o rekordy klasy HINFO (patrz str. 12.2.7) (informacje o komputerze: sprzęt, oprogramowanie),

`-M` pyta o rekordy klasy MX (patrz str. 12.2.4) (adresy komputerów mogących przejmować pocztę w razie awarii komputera),

`-N` pyta o rekordy klasy NS (patrz str. 12.2.2) (adresy baz danych wiążących adresy internetowe z nazwami),

`-P` pyta o rekordy klasy PTR (patrz str. 12.2.6) (wskaźniki pozwalające na podstawie numeru internetowego znaleźć nazwę komputera),

`-S` pyta o rekordy SOA (patrz str. 12.2.1) (adres osoby odpowiedzialnej za zarządzanie domeną),

`-T` pyta o rekordy TXT (patrz str. 12.2.9) (dodatkowe informacje o komputerze),

`-W` pyta o rekordy WKS (patrz str. 12.2.8) (usługi oferowane przez komputer),

`-*` pyta o wszystkie informacje,

`-a` próbuje wszystkich dostępnych metod (wyklucza użycie opcji `-s`),

`-s` *`server`* pragnie uzyskać informacje ze wskazanego komputera (zawierającego bazy danych),

host1 ... hostn nazwy (lub adresy internetowe) komputerów, o których chcemy uzyskać informacje,

`-?` wyświetla krótką informacje o programie,

`-version` informuje o wersji programu.

Rozdział 6

Połączenie „terminalowe” z innym komputerem

Do łączenia się użytkownika ze zdalnym komputerem służą komendy **telnet** i **rlogin**. Można również wykorzystać program **kermit** (patrz str. 11). Mimo iż najczęściej jest on wykorzystywany do połączeń terminalowych z wykorzystaniem modemu w najnowszych wersjach można z powodzeniem wykorzystywać go do uzyskiwania połączeń z wykorzystaniem sieci.

Funkcja zdalnego terminala umożliwia lokalnemu użytkownikowi (po połączeniu się ze zdalnym komputerem) na wydawanie komend interpretowanych przez zdalny system operacyjny i uruchamianie aplikacji. Bardzo ważną sprawą jest właściwe poinformowanie komputera, z którym się łączymy, jakie funkcje pozwala realizować używany terminal. Pozwoli to zdalnym aplikacjom w odpowiedni sposób pozycjonować kursor i właściwie interpretować naciskane przez nas klawisze.

Cechą specyficzną programów **telnet** i **rlogin** w systemie UNIX jest to, że nie emulują one (na ogół)[1] żadnego typu terminala. Służą one jedynie do możliwie „przeźroczystego” transportu znaków między komputerem zdalnym i lokalnym. Oznacza to tyle, że jeżeli pracujemy na komputerze lokalnym z terminala określonego typu, łącząc się z innym komputerem ciągle będziemy pracować z terminala tego samego typu. Cały problem polega jedynie na poinformowaniu zdalnego komputera o typie terminala z którego pracujemy. Protokól połączenia przekazuje podstawowe informacje o typie terminala. Problem może polegać na tym, że taki typ nie jest rozpoznawany przez zdalny komputer albo na tym, że jest źle ustawiony na komputerze lokalnym.

Co robić w takim przypadku?

Niestety nie ma no to pytanie łatwej i jednoznacznej odpowiedzi. Musimy posiadać pewną minimalną wiedzę na temat:

- jak inaczej możemy nazwać typ terminala z którego pracujemy (na przykład łącząc sie z komputera mikro-vax jest bardzo prawdopodobne, że uzgodni on z komputerem zdalnym terminal typu `dec-vt100`, co nie zostanie najprawdopodobniej zrozumiane; możemy terminal przedefiniować na `vt100`; podobnie, pracując z komputera HP z terminala `hpterm` – niezbyt często rozumianego przez inne komputery – zastępujemy go terminalem typu `hp`).
- jakie typy terminala rozpoznaje komputer, z którym się łączymy (można wylistować kartotekę `/usr/lib/terminfo` wraz z podkartotekami – na przykład komendą `ls -R /usr/lib/terminfo | more`).

Warto również w zbiorze `.profile` (jeżeli korzystamy z **ksh** lub **sh** i pochodnych) zdalnego komputera umieścić linię o takiej lub podobnej treści ([16]):

```
eval ‘tset -m :\?${TERM:-vt100} -r -s -Q‘
```

Dla **csh** może wyglądać tak (**???????**)

```
set noglob
eval ‘tset -s -Q -m ’?vt100’‘
```

Komenda ta odczytuje ze zmiennej środowiska typ terminala uzgodniony podczas nawiązywania połączenia; jeżeli zmienna środowiska jest pusta podstawia za nią terminal typu „vt100” (można tu wstawić typ bardziej nam odpowiadający), a następnie odpytuje użytkownika czy wartość jest właściwa:

[1] Wyjątkiem jest program **tn3270** służący do komunikacji z komputerami IBM.

```
TERM = (vt100)
```

na koniec, wreszcie, ustawia odpowiedni typ terminala w środowisku i inicjalizuje go.

Po połączeniu – możemy sprawdzić wartość zmiennej środowiska `TERM` za pomocą komendy **env** lub wydając polecenie:

```
echo $TERM
```

a następnie próbować zmienić jej wartość (gdy jest niewłaściwa) za pomocą komend:

`TERM=`*`value`*`; export TERM` – gdy używamy **sh** lub **ksh**
`setenv TERM` *`value`* – gdy pracujemy z **csh**.

Kolejny problem, który może sprawiać problemy ustalenie zestawu znaków używanych do:

- kasowania poprzedniego znaku,
- przerwania wykonywania procesu,
- zawieszenia procesu.

Do sprawdzenia i zmiany wartości tych parametrów w systemie Unix służy komenda **stty**. Odpowiednie ustawienia możemy sprawdzić wydając komendę: `stty -a`, a żądane wartości ustawiać w pliku `.profile` lub `.login` albo wydawać z linii komendy.

Najczęściej spotykane ustawienia parametrów: kasowanie poprzedniego znaku (erase): Ctrl–H lub Del; przerwanie procesu (): Del albo Ctrl–C; zawieszenie wykonania zadania (susp): Ctrl–Z.

Można zdefiniować jeszcze (jeżeli nie zostało to zrobione) znak służący do kasowania słowa (werase): Ctrl–W i całej linii (kill): Ctrl–U.

Odpowiedni zestaw znaków należy wybrać w zależności od przyzwyczajeń użytkownika i możliwości terminala, z którego nawiązuje połączenie.

Oto przykładowe ustawienie parametrów (wygodne podczas pracy z terminala **hpterm**):

```
export TERM=hp
stty erase \^H intr \^C
```

W dawnych czasach (gdy pracowano na terminalach typu *teletype* – dalekopis) znakami używanymi do kasowania znaku poprzedniego był: `#`, a znakiem używanym do kasowania całej linii: `@`. Niestety, te czasy nie odeszły w przeszłość tak zupełnie: znaki te są używane przez system HP–UX jako znaki standardowe i zmieniane komendą **stty**. W szczególności obowiązują one w trakcie procedury podawania nazwy użytkownika i hasła (z wszelkimi tego konsekwencjami).

6.1 Komenda telnet

6.1.1 System Windows 95

Rozdział ten jest jednym z kilku (inne opisują programy **ftp** (str. 7.7), **tracert** (str. 4.4.1) i **ping** (str. 4.3.1)) opisujących podstawowe usługi sieciowe dostępne w systemie Windows 95.

System Windows 95 wyposażony został w bardzo prostą (ichyba jedyną) aplikację „okienkową” umożliwiającą połączenie terminalowa z innym komputerem. Program jest bardzo prosty:

- pozwala na emulację dwu typów terminala:[2]
  - vt52,
  - vt100 (ansii).[3]
- pozwala na symboliczny wybór numeru portu z którym chcemy się łączyć:
  - telnet,
  - daytime,
  - echo,

[2] Niestety, wielu specjalistów uważa, że jest to **bardzo** zła emulacja.

[3] W bardzo wielu komputerach Unixowych właściwości terminali vt100 i ansii różnią się nieco. Z moich doświadczeń wynika, że nieco lepiej sprawdza się emulacja vt100, przy czym komputer z którym si łączymy powinien wiedzieć, że terminal jest typu vt100 a nie ansii.

  - quotd,
  - chargen;

- określić typ terminala, jakim program będzie się „przedstawiał” serwerowil; do wyboru mamy:
  - `vt100`,
  - `VT100`,
  - `ansii`,
  - `ANSII`,
  - `DEC-VT100`;

i to właściwie wszystko.

Program można uruchamiać na dwa sposoby:

1. odnajdując jego ikonę w kartotece `Windows`,
2. wydając polecenie `telnet [`*adres* `[`*port*`]]` – w linii polecenia może pojawić się adres komputera (i, opcjonalnie, port), z którym się łączymy.

6.1.2 System UNIX

Komenda **telnet** wywoływana jest w następujący sposób:
`telnet [`*host* `[`*port*`]]`

Protokół TELNET wykorzystywany do komunikacji ze zdalnym komputerem jest bardzo mocno rozbudowany, program **telnet** pozwala korzystać z wielu jego możliwości. Do normalnej pracy nie trzeba na szczęście znać wszystkich możliwości tego programu. Tu podamy jedynie najbardziej podstawowe.

W lini komendy mogą pojawić się dwa parametry:

host nazwa lub adres IP komputera z którym chcemy się połączyć, gdy nie zostanie podana program telnet zgłosi się w następujący sposób:

```
telnet>
```

i będzie oczekiwał na podanie komend (o tym nieco później).

port jeżeli występuje pierwszy parametr, może być podany drugi, określający z którym „portem” (patrz str. 3.2) odległego komputera telnet ma się łączyć.

Podczas nawiązywania połączenia, uzgadnianych jest szereg parametrów mających zapewnić odpowiednią „przeźroczystość” kanału przesyłania. Nie będziemy ich omawiać odsyłając czytelnika do odpowiedniej dokumentacji: strony manuala (**telnet**(1)) i odpowiedniego RFC.[4]

Po nawiązaniu połączenia powinien poawić się napis podobny do tego:

```
Connected to ldscoux.immt.pwr.wroc.pl

Escape character is ^]
```

i w chwilę później powinna pojawić się winetka zapraszająca do „zalogowania sie” na oddalonym komputerze.

Jeżeli komendę **telnet** użyjemy bez podania nazwy komputera, z którym chcemy się połączyć lub w trakcie połączenia użyjemy „*escape character*”[5] (standardowo znak ten uzyskuje się naciskając klawisz Ctrl i gdy jest on naciśnięty naciskając]) **telnet** przechodzi do trybu komend. W trybie tym podajemy komendy z klawiatury, przy czym wystarczy dla kazdej komendy podać tylko tyle znaków, aby odróżnić ją od innej. Oto spis najważniejszych komend:

`open` *host* `[`*port*`]` otwiera połączenie z zadanym komputerem i wskazanym portem;

`close` zamyka połączenie telnetowe; jeżeli połącenie nawiązane było z trybu komend – zostajem w tym trybie; w przeciwnym razie **telnet** kończy pracę;

`quit` zamyka połączenie i kończy pracę programu;

[4] Request For Comments – dokumenty będące swoistymi standardami sieci Internet.
[5] *escape* po angielsku znaczy ucieczka.

`z` zawiesza **telnet**; jeżeli program został uruchomiony z powłoki (*shell*) umożliwiającej sterowanie zadaniami (**ksh**, **csh**) – sesja zostaje zawieszona, a użytkownik wraca do powłoki, z której uruchomił **telnet**; wznowienie pracy następuje po wykonaniu komendy **fg** powłoki;

`status` pokazuje aktualny stan programu;

`display` wyświetla stan wszystkich lub niektórych parametrów ustawianych komendą `set` str. (6.1.2) i `toggle` (str. 6.1.2);

`? [`*`command`*`]` udziela informacji o wszystkich dostępnych komendach lub o komendzie wyspecyfikowanej;

`! [`*`shellcommand`*`]` pozwala uruchomić komendę lub „wyjść" na chwilę do powłoki;

`send` *`arguments`* wysyła jeden lub więcej specjalych sekwencji znaków do komputera, z którym jesteśmy połączeni; argumentami mogą być (w linii można wyszczególnić więcej niż jeden):

- `?` udziela informacji o dostępnych sekwencjach, które może wysłać lokalny **telnet**,
- `escape` wysyła do zdalnego komputera sekwencję *escape* – początkowo Ctrl–] może to mieć zastosowanie w sytuacji gdy z komputera A połączyliśmy się z komputerem B, a z niego (również za pomocą komendy **telnet**) z komputerem C; naciskając klawisze Ctrl-] sekwencję przejmuje telnet na komputerze A, żeby go przekazać telnetowi na komputerze B musuimy użyć tej komendy;
- `ayt` wysyła zapytanie *Are You There* do zdalnego komputera – pozwala to zorientować się (w przypadku wolnych połączeń na duże odległości) czy połączenie ze zdalnym komputerem jest ciągle aktywne,
- `ec` wysyła sekwencję „kasuj znak" – może to być przydatne gdy z powodu złych definicji znaków sterujących nie możemy skasować znaku,
- `ip` wysyła sekwencję przerwij wykonanie procesu,
- `el` wysyła sekwencję kasuj linię – może być łatwiej skasować całą linię i wprowadzić ją od nowa niż poprawiać ją znak po znaku (zwłaszcza w przypadku wolnych połączeń),

`set` *`argument value`* komenda, pozwalająca ustawić kilka parametrów pracy programu **telnet**; efektywne wykorzystanie tych możliwości wymaga dobrej znajomości protokołu TELNET – nie radzimy majstrować...

`toggle` *`arguments...`* komenda pozwalać kilka „przełączników" ustalających różne parametry protokołu TELNET – nie radzimy ich zmieniać bez jego dobrej znajomości...

6.1.3 MS–DOS i PC/TCP

Oprogramowanie firmy FTP Software pozwala emulować dwa zasadnicze rodzaje terminali (w obu przypadkach program nazywa się **tn**):

vt220 rodzina terminali (vt52, vt100, vt220) pracujących według standardów opracowanych w firmie DEC, bardzo popularne terminale, prawie każdy z systemów komputerowych rozpoznaje co najmniej terminal **vt100**,

IBM 3270 rodzina terminali pełnoekranowych typu **3270** wykorzystywanych w systemach hostów IBM, terminal ten posiada bardzo dużą „inteligencję" (wykonując na przykład wszystkie funkcje poprawiania w swojej pamięci); wymiana danych odbywa się jednak „całymi ekranami" co jest powolne w przypadku pracy na duże odległości przy obciążonej sieci lub przy połączeniu za pomocą wolnych linii komunikacyjnych.

Ciekawą cechą programu **tn** jest to, że może stosować do potwierdzania autentyczności użytkownika protokołu **kerberos**. Wersja oprogramowania dostępna w USA pozwala również na szyfrowanie danych przesyłanych podczas sesji (korzystając z tego protokołu). Z powodu ograniczń eksportowych na technologie szyfrowania – oprogramowanie to nie było praktycznie używane w Polsce.

Program **tn** (skrócona nazwa od **t**el**n**et) służy do emulacji terminali **vt220** i **3270** i może być wykorzystany do łączenia się z komputerami IBM pracującymi pod kontrolą systemu operacyjnego VM/CMS, komputerami DEC pracującymi pod kontrolą systemu VMS i komputerami pracującymi pod kontrolą systemu operacyjnego Unix oraz wielu innymi komputerami podłączonymi do sieci Internet.

Użycie:

`tn [-n] [-h] [-w] [-t` *`type`*`| -x` *`ttype`*`]` *`host port`*
`tn -?|-version`

Znaczenie parametrów:

`-?` wyświetla podstawowe informacje o sposobie użycia programu,

`-version` podaje numer wersji programu,

`-h` pozwala na pracę w innym trybie video (ma to wpływ na liczbę linii tekstu wyświetlanych na ekranie) i tak:

terminal DEC vt pozwala na prace z ekranem o 43 lub 50 liniach (w zależności od rodzaju karty graficznej ega/vga),

terminal IBM 3270 pozwala na prace z ekranem o 32 lub 43 liniach (w zależności od wybranego podtypu terminala i karty graficznej);

o liczbie wierszy której używa terminal trzeba poinformować system operacyjny; w przypadku systemu UNIX należy użyć komend: `LINES=` *`n`* `; export LINES` (**sh** i **ksh** i pochodne) lub `setenv LINES` *`n`* (**csh** i pochodne),

`-w` zezwala na pracę w trybie 27 lini x 132 znaki (jeżeli posiadana karta graficzna pozwala na to); modyfikuje to jedynie pracę w trybie **3270**, gdyż w trybie **vt220**, jeżeli program użytkowy zażąda przejścia w tryb 132 znaków i karta graficzna na to zezwala to żądanie zostanie zrealizowane,[6]

`-n` wyłącza tryb rozszerzonych atrybutów terminala **3270**; rozszerzone atrybuty wymagają dodatkowych informacji co wydłuża czas transmisji,

`-t` *`type`* ustala typ terminala jaki ma emulować program; można wybierać z następujących typów: vt, vt220, vt100, vt52, 3277, 3278, 3279; dokładny typ terminala będzie negocjowany we wstępnej fazie połączenia z serwerem,

`-x` *`ttype`* ustala typ terminala jaki ma emulować program; można wybierać z następujących typów: vt, vt220, vt100, vt52, 3277, 3278, 3279; dokładny typ terminala **nie** będzie negocjowany we wstępnej fazie połączenia z serwerem,

host nazwa lub adres IP komputera, z którym chcemy się połączyć,

port numer portu (bramy) przez który chcemy się komunikować z komputerem; każdy z portów związany jest z jakąś funkcją realizowaną przez komputer;[7] opcja jest przydatna podczas usuwania błędów serwera.

Podczas pracy z programem **tn** można przejść do trybu wydawania poleceń temu programowi. Mastępuje to po naciśnięciu klawiszy Alt–F10.[8] Pojawia się wówczas (w najniższej linii ekranu) zapytanie „Command:". W odpowiedzi na to zapytanie możemy wydawać szereg jednoznakowych komend sterujących działaniem programu **tn**.

? wyświetla podpowiedzi o wszystkich dostępnych komendach,

! wywołuje **command.com** (bez zamykania sesji telneta) powrót do połączenia ze zdalnym komputerem następuje po wydaniu komendy **exit**.

Podczas pracy z DOSem serwer FTP jest zawieszony. Jeżeli w trakcie transmisji pliku wywołaliśmy DOSa, transmisja jest wstrzymana. Zostanie wznowiona po powrocie do pracy z telnetem. Zdalny klient może uznać to za przerwanie transmisji i przerwać połączenie!

a **tn** wysyła zapytanie *„Are You There?"* („Czy tam jesteś?"). Większość serwerów odpowiada wypisując na ekranie napis [Yes]. Jeżeli serwer nie odpowie mamy podstawy aby przypuszczć, że połączenie zostało przerwane.

b wysyła specjalny pakiet żądający przerwania wykonywanego właśnie procesu (bez przerywania sesji),

c zamyka połączenie w sposób „kulturalny",

i czyta zawartość wskazanego zbioru, tak jak z klawiatury,

[6] Zwracam uwagę, że praca w trybie 132 znaków w linii wymaga na ogół odpowiedniego monitora!

[7] Na przykład port 13 związany jest z funkcją udzielającą informacji o aktualnej godzinie

[8] Ten sam efekt na komputerze pracującym pod kontrolą UNIX osiągamy zazwyczaj naciskając klawisze Ctrl–[. Zwracam jednak uwagę, że komendy UNIXowego **telneta** są nieco inne!

l włącza tryb lokalnego echa (normalnie każdy naciskany klawisz wysyła kod znaku, który jest odsyłany przez zdalny komputer wypisując w efekcie na ekranie odpowiednią literę),

o powoduje zapisywanie wszystkich informacji pojawiających sie na ekranie do zbioru o podanej nazwie,

q zamyka połączenie w sposób „brutalny”,

r włącza tryb zdalnego echa (patrz również l),

t wysyła sygnał przerwania,

0,1,...,9 tworzy kolejne połączenie ze zdalnym komputerem (tym samym, z którym jesteśmy połączeni lub innym); służy również do przełączania pomiędzy połączeniami,[9]

F włącza i wyłącza serwer FTP,

I wyświetla adres IP komputera, z którego uruchamiamy **tn**, w linii ststusu,

p pyta o numer strony kodowej DOS używanej do wyświetlania znaków (nie akceptuje strony 852),

s wysyła następujący po nim znak tylko do DOSa (**tn** przejmuje pełną kontrolę nad klawiaturą i ta komenda pozwala wysłać znaki do DOSa; jest to istotne gdy zainstalowane są jakieś programy rezydentne, z którymi użytkownik może komunikować się za pośrednictwem klawiatury),

u wyłącza linię statusu,

U włącza linię statusu,

W przełącza tryb ochrony przed zapisem serwera FTP, po włączeniu ochrony przed zapisem użytkownicy FTP mogą jedynie czytać dane z komputera, na którym uruchomiony jest program **tn**,

z wysyła pakiet nakazujący zdalnemu serwerowi zaprzestać wysyłania listingów produkowanych przez zdalną aplikację (nie przerywając jej działania),

Enter przełącza do następnej aktywnej sesji,

Ctrl–b włącza/wyłącza tryb binarny w trybie emulacji terminala DEC vt,

Ctrl–d wyświetla podstawowe informacje o stanie oprogramowania PC/TCP,

Ctrl–h wyświetla podpowiedzi o komendach wyświetlających informacje o stanie oprogramowania PC/TCP,

Ctrl–k wyświetla informacje konfiguracyjne,

Ctrl–l odtwarza pierwotną zawartość ekranu,

Ctrl–n wyświetla informacje o stanie sieci,

Ctrl–s wyświetla informacje o aktywnych połączeniach,

Ctrl–t wyświetla informacje o aktywnych sesjach.

komendy dostępne w trybie DEC vt100

B powoduje że klawisz *backspace* wysyła kod backspace (ASCII 8, ^H), naciśnięcie klawisza CTRL–*backspace* wysyła kod *del* (ASCII 127),[10]

d powoduje wyłączenie trybu łamania lini dłuższych niż szerokość ekranu, znaki powodujące przepełnienie linii będą wypisywane na ostatniej pozycji,

D powoduje że klawisz *backspace* wysyła kod *del* (ASCII 127), naciśnięcie klawisza CTRL–*backspace* wysyła kod backspace (ASCII 8, ^H),

[9] Liczba równocześnie otwartych połączeń zależy od parametrów konfiguracyjnych oprogramowania PC/TCP; normalnie wynosi 4–5.

[10] Klawisze *backspace* i *del* używane są w różny sposób na różnych komputerach; jedna z możliwości jest taka: *backspace* kasuje znak na lewo od kursora, *del* przerywa działanie aktualnie wykonywanego programu; inne może być takie: *del* kasuje znak na lewo od kursora, *backspace* nie ma żadnego specjalnego znaczenia; liczba możliwych kombinacji jest duża! Komenda **stty** systemu UNIX pozwala dostosować środowisko zdalnego komputera do własnych upodobań.

e przełącza telnet w tryb wysyłania każdego znaku przekazywanego z klawiatury (obowiązuje z domniemania),

E przełącza telnet w tryb wysyłania całej linii (po naciśnięciu klawisza Enter), pozwala to dokonywać poprawek lokalnie i może przyśpieszyć komunikację po wolnych liniach, na duże odległości; tryb ten wymaga włączenia lokalnego echa, nie może być stosowany z aplikacjami pracującymi w trybie interakcyjnym i reagującymi na pojedyńcze naciśnięcia klawiszy (edytory),

m pozwala zmienić jeden tryb emulacji terminala **vt** na inny (program przedstawia pięć dostępnych trybów i oczekuje na wybranie właściwego),

N powoduje, że klawisz Enter wysyła sekwencję znaków CR LF (wartość domyślna),

R powoduje, że klawisz Enter wysyła sekwencję znaków CR NUL,

w włącza tryb łamania wierszy dłuższych niż szerokość ekranu.

W trybie emulacji terminala **3270** dostępne są dwie komendy:

y znaki NUL w niewypełnionych polach pozostają bez zmiany,

Y znaki NUL w niewypełnionych polach zamieniane są na spacje.

6.2 Komenda rlogin

Komenda **rlogin** (i dodatkowo **remsh** (str. 6.2.1)[11] oraz **rcp** (str. 6.2.1)) zostały opracowane aby ułatwić życie użytkownikom komputerów (kiedyś głównie UNIXowych) którzy mają konta na wielu komputerach. Pozwalają one bardzo łatwo przenosić się z jednego komputera na drugi, zdalnie wykonywać komendy na oddalonym komputerze i kopiować pliki z komputera na komputer.

Od czasu opracowania komend (jak się je nazywa) „**r**" opracowano kilka innych środowisk pracy uniezależniających użytkownika od komputara, na którym rozpoczyna pracę; są to między innymi **VHE**[12] lub **DCE**[13]. To ostatnie ma szanse stać się standardem, gdyż do jego realizacji włączyły się najpoważniejsze firmy softwareowe.

Mimo, że możliwe są implementacje komend protokołu „**r**" na komputerach PC, nie zaleca się ich wykorzystywania. Idea protokołu opiera się na pojęciu „równoważnych komputerów".[14] Równoważność komputerów zapewnia się na dwa sposoby:

1. na poziomie komputerów – w specjalnym zbiorze wpisuje się adresy tych komputerów, które uważa sie za zaufane,
2. na poziomie użytkowników – użytkownik w specjalnym zbiorze wpisuje nazwy komputerów (i nazwy użytkowników), których uznaje za „zaufanych".

Zaufani użytkownicy (z obdarzonych zaufaniem komputerów) otrzymują prawo wykonywania komend, rozpoczynania sesji i kopiowania plików z pominięciem procedury sprawdzenia identyczności (podania nazwy użytkownika i ważnego hasła). Zakłada się, że procedura potwierdzenia identyczności wykonana na jednym komputerze nie musi być ponawiana na kolejnych.

I mimo, że komputer PC może być uznany za „zaufany" – robienie tego jest dużym nieporozumieniem. Nie można bowiem uznać, że stosowane tam zabezpieczenia są wystarczające (tym bardziej, że bardzo łatwo w wielu wypadkach przekonfigurować jeden komputer PC tak, że będzie widziany jako stacja robocza o zupełnie innym adresie).

Generalnie, komendy „**r**" uważane są za słaby punkt bezpieczeństwa komputerów i dlatego należy używać ich bardzo „ostrożnie".

Wielu administratorów zwracających uwagę na na bezpieczeństwo swoich komputerów, a potrzebujących często „przenosić" się z jednego komputera na drugi stosuje specjalną ..odmianę" komendy **rsh**: **ssh**. Wymaga ona zainstalowania specjalnego oprogramowania tak po stronie klienta jak i serwera, ale zapewnia znacznie wyższy poziom bezpieczeństwa. Dane przesyłane między komputerami są szyfrowane, opracowano też znacznie lepsze metody potwierdzania „tożsamości" komunikujących się ze sobą komputerów.

[11] W niektórych systemach nosi ona nazwę **rsh** (SunOS), a jeszcze w innych **rcmd** (SCO)
[12] Virtual Home Environment.
[13] Distributed Computed Environment.
[14] Z angielska *equivalent hosts*.

6.2.1 System UNIX

rlogin

Najbardziej podstawowe użycie komendy **rlogin** jest następujące:
`rlogin` *`rhost`* `[-e` *`c`*`] [-l` *`username`*`]`

Komenda ta pozwala na rozpoczęcie sesji na zdalnym komputerze z pominięciem normalnej procedury identyfikacji (jeżeli tylko zostały spełnione odpowiednie warunki – patrz informacje o pliku `.rhosts` (str. 6.2.1)). Najwygodniejsz sytuacja jest wtedy, gdy nazwa użytkownika na wszystkich komputerach jest taka sama. W przeciwnym przypadku należy ją podać po parametrze `-l`.

Kolejny parametr: `-e c` określa „*escape character*” – znak służący do zwrócenia uwagi programowi **rlogin**, że kolejny znak(i) chcemy przekazać programowi, a nie zdalnemu komputerowi. Znakiem tym normalnie jest tylda „(˜)”. Kolejne znaki jakie można podawać po znaku ucieczki, to:

`.` (kropka) żądanie przerwania połączenia,

`!` (wykrzyknik) umożliwia wykonanie komendy („shell escape”) na lokalnym komputerze,

`^Z` zawiesza wykonanie komendy **rlogin** i umożliwia użytkownikowi prcę na lokalnym komputerze; powrót do sesji za pomocą komendy **fg**,

`^Y` podobnie jak `^Z`, ale zawieszana jest praca tylko komendy **rlogin**,

`~` (tylda) przesyła znak tylda do zdalnego komputera.

rhost to oczywiście nazwa zdalnego komputera, z którym się łączymy (powinna być ona podana zgodnie z ogólnymi zasadami).

Połączenie z wykorzystaniem komendy **rlogin** jest tak „przeźroczyste” jak tylko to możliwe, wartość zmiennej środowiska `TERM` (jak i szybkość transmisji – ważne przy połączeniach z terminala podłączonego linią szregową) są transmitowane do zdalnego komputera.

remsh

Komenda **remsh** pozwala na wykonanie zadania na odległym komputerze, przy czym wyniki produkowane przez zadanie będą wyświetlane na lokalnym terminalu (podobnie, z lokalnej klawiatury będą pobierane dane wejściowe). Uruchamianie komend pełnoekranowych (takich jak np. edytor **vi**), jest niestety utrudnione (lub nawet niemożliwe).

Użycie komendy jest następujące:
`remsh` *`host`* `[-l` *`username`*`] [-n]` *`command`*

remsh łączy się z oddalonym komputerem *host* (nie jest dokonywana żadna procedura identyfikacji użytkownika (parz: informacje o pliku `.rhots` (str. 6.2.1))) i uruchamia komendę *command*; normalnie komenda wykonywana jest, tak jak bdyby wydał ją użytkownik wydający komendę **remsh** chyba, że określono innego użytkownika za pomocą parametru `-l`. Wykonanie komendy kończy się gdy uruchamiany program zakończy pracę.

Jeżeli nie wyspecyfikowano komendy, która ma być uruchomiona, używana jest komenda **rlogin** (str. 6.2.1) w celu uzyskania połączenia z oddalonym komputerem.

rcp

Nazwa komendy pochodzi od słów *remote file copy* – co można przetłumaczyć jako zdalne kopiowanie plików. Pozwala ona na kopiowanie plików (lub całych struktur kartotek) z jednego komputera na drugi (z pominięciem normalnych procedur identyfikacji użytkownika). Użycie komendy jest następujące:
`rcp [-p]` *`file1 file2`*
`rcp [-p] [-r]` *`file1`* `[`*`file2`* `...]` *`directory`*

rcp służy do kopiowania plików pomiędzy różnymi komputerami. Każdy z argumentów *file* lub *directory* jest bądź nazwą pliku zdalnego, o postaci *hostname:path* lub plikiem lokalnym. Gdy *hostname* ma postać *ruser@rhost* na oddalonym komputerze używana jest zamiast nazwy użytkownika nazwa *ruser*.

Gdy użyjemy modyfikatora `-p` **rcp** stara się zachować datę modyfikacji i prawa dostępu do kopiowanych plików. Gdy natomiast podamy `-r` i *file1* jest kartoteką, a *directory* jest nazwą istniejącej kartoteki, **rcp** tworzy w niej kartotekę o naziwe *file1* i kopiuje całą zawartość kartoteki źródłowej (wraz z podkartotekami). Gdy *directory* nie istnieje zostanie utworzona.

Możliwe są również operacje tego typu, żeby będąc na komputerze A kopiować pliki z komputera B na komputer C:
`rcp` *ruser1@rhost1:path1 ruser2@rhost2:path2*
przy czym operacja wykonywana jest w sposób następujący:
`remsh` *`rhost1`* `[-l]` *`ruser1`* `rcp` *`path1 ruser2@rhost2:path2`*
ze wszystkimi tego konsekwencjami.[15]

Komenda **rcp** nie pyta o hasło! Aby użytkownik miał prawo dostępu do plików na zdalnych hostach musi być odpowiednio wypełniony plik `.rhosts`.

Plik `.rhosts`

6.2.2 MS–DOS i PC/TCP

Program **rloginvt** służy do nawiązania połączenia z komputerem pracującym pod kontrolą systemu operacyjnego Unix. Wykorzystuje on specjalny protokół pozwalający (po odpowiednim skonfigurowaniu środowiska Unix) łączyć się z komputerem bez procedury przedstawiania się i podawania hasła.[16]

Dla porządku podamy podstawowe zasady użycia programu **rloginvt**.

Użycie:

`rloginvt [-d] [-43|-50]` *`host`* `[`*`user`*`]`
`rloginvt -?|-version`

Znaczenie parametrów podawanych w linii komendy jest następujące:

`-d` program wyświetla dodatkowe informacje diagnostyczne,

`-43|-50` przełącza terminal w tryb 43 (50) linii (pod warunkiem, że karta graficzna na to pozwala),

host nazwa lub adres IP komputera, z którym się łączymy,

user nazwa użytkownika (jeżeli inna niż w zbiorze konfiguracyjnym).

Po naciśnięciu klawisza Alt–F10 (podobnie jak w przypadku **tn**) można wydawać specjalne polecenia. Są one bardzo podobne do komend programu **tn**. Dotyczy to komend: c, q, !, ?, B, d, D, F, i, o, m, u, U, W, w.

[15] To *ruser1* musi mieć prawo dostępu do plików użytkownika *ruser2* na komputerze *hosts2*.

[16] Zwracamy uwagę, że nie jest to specjalnie bezpieczne, w bardzo wielu przypadkach z komputera PC może skorzystać przypadkowy użytkownik.

Rozdział 7

Przesyłanie zbiorów – file transfer protocol

Aby przesłać zbiory z jednego komputera na drugi należy zestawić połączenie logiczne pomiędzy nimi. Program biernie oczekujący na realizacje funkcji nazywany jest *serwerem*; czynną rolę odgrywa *klient*, nawiązuje on połączenie i służy jako pośrednik (interface) pomiędzy użytkownikiem a zdalnym komputerem.

Program realizujący funkcje serwera nazywa się najczęściej **ftp** (i jest częścią praktycznie każdego zestawu oprogramowania sieciowego). Opracowano kilka różnych implementacji serwera: **ncftp** o wielu ułatwiających życie funkcjach czy X–Windowsowe interfejsy pozwalające na bardzo wygodne wskazywanie myszą plików, które chcemy kopiować. W środowisku MS Windows dostępne są również w pełni graficzne interfejsy protokołu FTP. Możliwości i zestaw funkcji realizowany przez klientow FTP jest bardzo różny i należy pogodzić się z faktem, że nie będziemy w stanie opisać ich wszystkich.

Na komputerach wyposażonych w wieloprogramowe i wielodostępne systemy operacyjne (VMS, UNIX, MS–Windows…) serwer (w uśpieniu) oczekuje na nawiązanie połączeń. Na komputerach jednoprogramowych (jak PC wyposażony w system operacyjny MS DOS) serwer musi być uruchomiony specjalnie (blokując inne wykorzystanie komputera). Bardzo często realizacje programów **telnet** realizują oprócz funkcji „klient telnet” funkcję „serwer ftp” – pozwala to na równoczesne przesyłanie plików i pracę interakcyjną na komputerze wielodostępnym (o ile implementacja programu **telnet** pozwala na otwarcie kilku sesji).

7.1 Nawiązanie połączenia

Aby nawiązać połączenie z komputerem, na którym pracuje *serwer* ftp należy wydać polecenie:

`ftp` *server–host*

server–host powinna byc oficjalną nazwą komputera, lub jego adresem numerycznym (w postaci `n.n.n.n`). Czas potrzebny na uzyskanie połączenia zależy od odległości pomiędzy komputerami, obciążenia sieci i obciążenia komputera z którym łączymy się.

Po uzyskaniu połączenia serwer przedstawia się i prosi o podanie nazwy użytkownika. Prawo do korzystania z usługi **ftp** mają wszyscy zarejestrowani użytkownicy komputera, z którym się łączymy. Przedstawiają się oni i podają własne hasło. Mają dostęp do wszystkich plików systemu (z ograniczeniami wynikającymi z praw dostępu).

Niektóre komputery udostępniają usługę zwaną *anonymous ftp*. Użytkownik pragnący z niej skorzystać przedstawia się jako *anonymous*.[1] Jako hasło należy podać swój adres poczty elektronicznej w postaci np.: user@host.domain.[2] Bardzo często można podać jako hasło użytkownika anonimowego cokolwiek. Niektóre serwery sprawdzaja jednak czy podany adres zwrotny zgodny jest z nazwą komputera, z którego dokonujemy połączenia!

Adres zwrotny bywa wykorzystywany przez bardzo porządne archiwa publiczne oprogramowania do informowania użytkowników o odkrytych błędach, wirusach itp. w skopiowanym oprogramowaniu.

Wszystkie połączenia są rejestrowane! Większość komputerów udostępnianych jako serwery *anonymous ftp* wykorzystywana jest również do normalnych, statutowych, zadań. Dlatego też należy starać się nie kontaktować z nimi w godzinach normalnej ich pracy (zwykle od 8:00 do 16:00 – 18:00 lokalnego czasu).

[1] Większość komputerów dopuszcza również przedstawienie się jako *ftp*.
[2] W bardzo wielu przypadkach dobrym rozwiązaniem jest podanie swego nazwiska zakończonego znakiem „@”

Użytkownik *anonymous* posiada prawa gościa, to znaczy dosęp do wydzielonych obszarów systemu plików. Część serwerów pozwala na przyjmowanie zbiorów przekazywanych przez anonimowych użytkowników. Zapisywane są one zwykle w kartotece `/incoming`.

Należy pamiętać, że usługa *anonymous ftp* jest **przywilejem** społeczności sieciowej; w żadnym wypadku nie jest **obowiązkiem** administratorów komputerów. W przypadku utrudnień stwarzanych przez gości – usługa może być ograniczona lub zlikwidowana! Z przywilejów należy korzystać rozsądnie i z umiarem!

7.2 Przeglądanie zawartości archiwum

Komendy `dir` lub `ls` służą do przeglądania zawartości systemu plików archiwum. Komenda `dir` podaje nazwy zbiorów, prawa dostępu do nich, ich długość. Format wydruku zależy od systemu operacyjnego komputera, z którym się łączymy; na komputerze UNIXowym jest zgodny z formatem oferowanym przez komendę `ls -l`. Komenda `ls` podaje często skrócony wydruk zawartości katalogu (bez podawania długości zbiorów); nie jest to jednak regułą i zależy od właściwości serwera. Bardzo często komendy `ls` i `dir` są równoważne.

Poniżej podajemy przykładowy wygląd sesji i wydruk zawartości archiwum na komputerze UNIXowym (wraz z objaśnieniem znaczenia poszczególnych kolumn wydruku):

```
(hp)[1026]myszka> ftp ftp.immt.pwr.wroc.pl

Connected to ldhpux.immt.pwr.wroc.pl.
220 ldhpux FTP server (Version 1.7.193.3 Thu Jul 22 18:32:22 GMT 1993) ready.
Name (ftp.immt.pwr.wroc.pl:myszka): ftp
331 Guest login ok, send ident as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 78
-rw-r--r--   1 root     3           357 Jul 21 11:25 .Links
-rw-r--r--   1 ftp      22         1323 Nov 20 16:25 .cache
-rw-r--r--   1 ftp      22         4536 Nov 13 14:16 .cache+
-rw-r--r--   1 ftp      3           795 Jul 21 11:33 .names
drwxr-xr-x  46 root     3          2048 Jul 21 11:25 Gopher_Jewels
drwxr-xr-x   2 root     3          1024 Jul 21 11:25 bin
drwxr-xr-x   2 root     3            24 Jul 21 11:25 cdrom
drwxr-xr-x   3 root     3          1024 Jul 21 11:25 cdromy
drwxr-xr-x   2 root     3          1024 Jul 21 11:25 etc
drwxr-xr-x   2 201      3          1024 Jul 21 11:25 fff
drwx------   2 root     3            24 Jul 21 11:25 incoming
drwxr-xr-x   2 root     3          1024 Jul 21 11:25 info-ww
-rw-r--r--   1 root     3          1446 Jul 21 11:25 info.i-19
drwxr-xr-x   2 root     3          1024 Jul 21 11:25 kbn
drwxr-xr-x   2 root     3          1024 Jul 21 11:25 lib
-rw-r--r--   1 root     3          9961 Nov 19 02:19 ls-lR.Z
-rw-r--r--   1 root     3           945 Jul 21 11:25 odstep
drwxr-xr-x   3 root     3          1024 Jul 21 11:29 periodicals
drwxr-xr-x  34 root     3          1024 Jul 21 11:33 pub
drwxr-xr-x   2 root     3          1024 Jul 21 11:25 telefony
drwxr-xr-x   2 root     3          1024 Jul 21 11:25 test
drwxr-xr-x  16 201      3          1024 Sep 21 16:54 wroc
drwxr-xr-x  20 201      3          1024 Oct 12 12:05 www
226 Transfer complete.
ftp> quit
221 Goodbye.
```

Wydruk zawartości zdalnego archiwum składa się z siedmiu kolumn:

```
-rw-r--r--   1 root     3          9961 Nov 19 02:19 ls-lR.Z
```

```
|     1     | 2 | 3 |     4          | 5 |      6      |   7   |
```

Oto ich znaczenie:

1. Określa prawa dostępu i typ pliku. Na dziesięciu znakach zapisane są następujące informacje:
   - na pierwszej pozycji zapisany jest rodzaj pliku:

     `-` oznacza zwykły plik,

     `d` oznacza kartotekę,

     czasami spotkamy również:

     `l` odsyłacz („*link*") pozwalający na lepsze organizowanie struktury plików.
   - następne trzy znaki określają uprawnienia właściciela pliku (patrz rownież opis trzeciej kolumny); uprawnienia opisane są przez trzy literki `r`, `w`, `x` umieszczone na trzech kolejnych pozycjach; wystąpienie litery oznacza (kolejno) prawo do czytania, zapisu, wykonania pliku; brak uprawnienia zaznaczany jest wpisaniem myślnika `-` na odpowiedniej pozycji. Prawo wykonania oznacza prawo przeglądania (i wykonania komendy **cd**) w przypadku kartotek.
   - na kolejnych trzech znakach zapisane są analogicznie uprwanienia „grupy" (czyli użytkowników w jakiś sposób związanych z właścicielem pliku) – patrz również opis kolumny czwartej,
   - na ostatnich trzech znakach zapisane są uprawnienia wszystkich innych użytkowników systemu, ta informacja jest najważniejsza dla użytkownika korzystającego z anonimowego ftp.
2. Informacja niezbyt istotna dla zwykłego użytkownika; w przypadku kartoteki określa liczbę (plików, podkartotek) w niej zawartych; w przypadku zwykłego pliku pod iloma nazwami plik ten występuje.
3. Nazwa (lub numer) właściciela pliku. Każdy użytkownik systemu UNIXowego ma swoją nazwę – identyfikator (służy on do przedstawienia się podczas nawiązywania połączenia) oraz należy do jednej (lub więcej) grup użytkowników. Jest on właścicielem każdego tworzonego przez siebie pliku. Dodatkowo z plikiem związana jest jedna z grup do której należy użytkownik.
4. Nazwa grupy. (Patrz również punkt 3.)
5. Długość pliku (w bajtach).
6. Data ostatniej modyfikacji pliku.
7. Nazwa pliku (wielkie i male litery są znaczące, tak samo jak i wszystkie inne znaki).

Zmiany kartoteki dokonujemy za pomocą komendy `cd`. Nazwę bieżącej kartoteki podaje komenda `pwd`. Przejśccie do kartoteki „wyższej" zależy od rodzaju systemu operacyjnego serwera: można próbować `cd ..`, ale dostępna jest komenda `cdup`.

7.3 Przesyłanie plików

Do przesyłania zbiorów służą komendy `get` (pojedyńczy zbiór z serwera do klienta), `put` (pojedyńczy zbiór w przeciwną stronę) oraz `mget` i `mput` pozwalające na przesyłanie wielu zbiorów.

Większość komputerów pracujących w sieci internet to komputery UNIXowe. Dlatego też należy zwracać uwagę na duże i małe litery. Są one zazwyczaj znaczące!

Zwracam uwagę, że różne komputery przechowują informacje w bardzo różnej formie (zazwyczaj innej niż przywykliśmy na komputerach PC). Dotyczy to zwłaszcza zbiorów tekstowych. Podczas przesyłania informacji muszą/mogą być dokonywane różne rodzaje konwersji. Program **ftp** wyposażony jest w odpowiednie mechanizmy zapewniające odpowiednią „czytelność" zbiorów po przesłaniu. Dotyczy to zwłaszcza zbiorów tekstowych. Standardowo serwer i klient ftp są ustawione w takim trybie, aby prawidłowo przesyłać zbiory tekstowe. Komenda `binary` powoduje wyjście z tego trybu do trybu przesyłania, w którym nie dokonuje się żadnych konwersji transmitowanych danych. Powrót do pierwotnego trybu następuje po wydaniu komendy `ascii`.

Jak rozróżnić rodzaj zbioru? Nigdy nie będziemy mieli całkowitej pewności, ale patrząc na nazwę można wysnuć pewne wnioski:

- zbiory o rozszerzeniach `Z`, `z`, `gz`, `zip`, `arc`, `zoo`, `arj`, `exe`, `tar`, `dvi` są zbiorami binarnymi,
- zbiory o rozszerzeniu `doc`, `txt`, `tex`, `lst`, `prn`, `sh` będą zazwyczaj zbiorami tekstowymi,

- zbiory o nazwie `read.me`, `readme` lub podobnej (zachęcającej do czytania) są, prawie na pewno, zbiorami tekstowymi.

Jeżeli zdarzy się, że prześlemy zbiór tekstowy (z komputera UNIXowego) w trybie binarnym, to nie ma tragedii. Program **unix2dos** (lub podobny) pozwala przekształcić zbiór tekstowy do postaci czytelnej dla DOSa. Jeżeli prześlemy zbiór binarny w trybie tekstowym – nie ma dla niego ratunku!

7.4 Plik `.netrc`

Program **ftp** systemu UNIX współpracuje z plikiem o nazwie `.netrc` znajdującym się w głównej kartotece użytkownika. Współpraca ta odbywa się w sposób typowy dla wielu aplikacji systemu UNIX, które w momencie startu czytają swój zbiór konfiguracyjny – którego nazwa zwykle kończy się na „`rc`".

Zbiór `.netrc` może zawierać informacje o sposobie „przedstawinia się" obcym komputerom (nazwa komputera, nazwa użytkownika, hasło). Informacje te mogą mogą być wykorzystane przez opcję „*autologin*" (str. 7.6).

Można w tym pliku również zdefiniować makra (własne komendy z parametrami) – patrz komenda `macdef` (str. 7.6). Makro o nazwie `init` będzie automatycznie wykonane po nawiązaniu połączenia z komputerem (procedura „*autologin*")

Plik `.netrc` może być wykorzystany do przechowywania informacji niezbędnych do zalogowania się jako „prawdziwy" użytkownik systemu. Wymaga to wpisania „rzeczywistego" hasła. Mimo, że jest to możliwe **nie należy** korzystać z tej możliwości! Ponieważ ludzie są tylko ludźmi, wiele implementacji programu **ftp** nie skorzysta z tego pliku, jeżeli jest on dostępny publicznie do czytania. Prawa dostępu należy ustawić wówczas dosyć restrykcyjnie za pomocą komendy:

```
chmod 0600 .netrc
```

7.4.1 Zawartość pliku `.netrc`

Plik `.netrc` jest interpretowany przez procedurę *autologin* (str. 7.6) i zawiera informacje umożliwiające automatyczne „zalogowanie" się na komputerze, z którym **ftp** nawiązuje połączenie.

W pliku `.netrc` mogą znajdować się oddzielone spacjami lub znakami tabulacji lub znakami nowej linii następujące komendy (słowa kluczowe ????):

`machine` *`name`* identyfikuje komputer którego zapis dotyczy, gdy nazwa zgadza się, interpretowane są kolejne zapisy;

`default` pozwala stworzyć zapis zgodny z nazwą każdego komputera (zastępuje `machine`);

`login` *`name`* określa nazwę użytkownika, jaką procedura *auto–login* przedstawi nas komputerowi, z którym się łączymy;

`password` *`string`* określa hasło jakie zostanie użyte w procesie łączenia się z komputerem. Podręczniki dotyczące bezpieczeństwa komputerów **nie zalecają** wpisywanie do pliku `.netrc` rzeczywistych haseł, radzą rownież by ograniczć prawa do odczytu tego pliku;

`account` *`string`* podaje dodatkowe informacje, które mogą być potrzebne aby mieć prawo dostępu do pewnych kartotek; porównaj komendę `account` (str. 7.6).

`macdef` *`name`* definiuje makro w sposób analogiczny do komendy `macdef` (str. 7.6); kolejne linie są zapamiętywane, aż do napotkania co najmniej jednej linii pustej; jeżeli zostało zdefiniowane makro o nazwie `init` zostanie ono automatycznie wykonane na zakończenie procesu *auto–login*.

Oto przykładowy zapis zbioru `.netrc` pozwalający łatwo łączyć się z jednym z archiwów:

```
machine ftp.tex.ac.uk login ftp password
        myszka@ldhpux.immt.pwr.wroc.pl
```

7.4.2 Przykładowy plik `.netrc` pozwalający na automatyczny transfer zbioru

Poniższy przykład pozwala na automatyczny transfer pliku z oddalonego archiwum anonymous ftp w godzinach mniejszego ruchu:

```
machine ftp.tex.ac.uk login ftp
password myszka@ldhpux.immt.pwr.wroc.pl
macdef init
cd pub/archives/support
bin
get file.name
quit
```

Teraz wystarczy uruchomienie programu **ftp** z jednym parametrem:

```
ftp ftp.tex.ac.uk
```

aby (o ile się to uda) automatycznie połączć się z komputerem i przesłać żądany plik.

7.5 Szczegółowe uwagi na temat korzystania z ftp

7.5.1 IBM VM/System

Współpraca z tym systemem za pomocą **ftp** jest nieco inna niż z systemami UNIXowymi. Połączenie wygląda tak:

```
ftp plwrtu11.ci-pwr.wroc.edu.pl
```

Po połączeniu uzyskujemy informację, że nie została zdefiniowana kartoteka robocza – nie będziemy mogli przesyłać plików. Komenda **cd** w odniesieniu do systemu VM jest żądaniem przyłączenia odpowiedniego dysku wirtualnego; dyski takie są definiwane przez nazwę użytkownika i sdres, odpowiednia komenda ma postać:
`ftp> cd` *login–name.disk–address*
i rozpoczyna połączenie z dyskiem użytkownika (o wskazanym adresie), dostęp do dysku zabezpiecza system haseł – należy użyć komendy `account` (str. 7.6)

Adresy dysków użytkownika podaje komenda `q disk` wydana podczas normalnej (telnetowej) sesji.

Nazwy plików w systemie VM składają się z dwu łańcuchów znaków (każdy o długości do ośmiu znaków), używając ich w komendzie `get` (str. 7.6) należy rozdzielić je kropką:

```
get all.notebook
```

7.5.2 System VMS firmy DEC

System plików na komputerach (mikro)VAX ma nieco inną strukturę niż systemy MS–DOS lub UNIX. Poruszanie się w niej może sprawiać pewne kłopoty.

Nazwa pliku składa się z trzech części: nazwy, rozszerzenia i numeru wersji. Nazwa od rozszerzenia oddzielona jest kropką, a numer wersji od rozszerzenia średnikiem. W bardzo wielu przypadkach możemu numerem wersji się nie przejmować: nazwa pliku podana bez numeru wersji równoważna jest najwyższemu istniejącemu numerowi wersji (przy odczycie) i numerowi wyższemu o jeden od najwyższego (przy zapisie) – lub jedynce, jeżeli plik nie istnieje.

Kartoteki (podczas listowania) mają rozszerzenie `DIR`

7.6 System UNIX

Program **ftp** uruchamia się za pomocą komendy:

```
ftp [-g] [-i] [-n] [-v] [server-host]
```

Znaczenie poszczególnych parametrów jest następujące:

- `-g` wyłącza funkcję zwaną z angielska „*globbing*"; normalnie funkcja jest włączona (komenda `glob` na stronie 7.6).
- `-i` wyłącza interakcyjne odpytywanie podczas przesyłania wielu zbiorów; porównaj komendę `prompt` (str. 7.6).
- `-n` wyłącza opcję „*auto–login*" (por. informacje o pliku `.netrc` (str. 7.4)).

`-v` włącza opcję szczegółowego informowania o przebiegu połączenia i transmisji („*verbose*”); por. komendę `verbose` (str. 7.6).

server–host nazwa komputera, z którym chcemy nawiązać połączenie – nie musi być podana w lini komendy **ftp**; jeżeli nazwa komputera jest podana – ftp natychmiast podejmuje próbę nawiązania połączenia; w przeciwnym razie oczekuje na podanie jej przez użytkownika (komendą `open`).

Poniżej podajemy zestaw komend rozumianych przez klienta **ftp**:

`! [`*`command`*` [`*`args`*`]]` wywołuje shell (na lokalnym komputerze); jeżeli wyszczególniono *command* wykonywana jest podana komenda i następuje powrót do programu **ftp**; w przeciwnym razie powrót do programu **ftp** następuje po wydaniu komendy `exit`.

`$ `*`macro-name`*` [`*`args`*`]` wykonuje zestaw komend (makro) zdefiniowane za pomocą komendy `macdef` (str. 7.6); argumenty są przekazywane do komendy „*unglobbed*”.

`account [`*`passwd`*`]` dostarcza dodatkowe hasło wymagane przez system, z którym się połączyliśmy; jeżeli nie podano argumentu użytkownik pytany jest o hasło. Komenda pozwala na takie zorganizowanie archiwum, aby chroniić dostęp do pewnych kartotek hasłem – nie tworząc osobnego konta dla użytkownika.

`append `*`local--file`*` [`*`remote--file`*`]` kopiuje zbiór *local–file* dodając go do końca zbioru *remote–file*; gdy nie podano drugiego parametru używana jest nazwa *local–file* (po ewentualnym dokonaniu transformacji wynikających z komend `ntrans` (str. 7.6) i `nmap` (str. 7.6)).

`ascii` ustawia tryb przesyłania plików na tekstowy (jest to tryb domniemany!).

`bell` powoduje, że po zakończeniu transferu **ftp** wydaje dźwięk.

`binary` ustawia typ przesyłania na binarny.

`bye` kończy połączenie ze zdlnym komputerem i prace programu; podobne dziaólnaie ma użycie znaku end–of–file (EOF) (bardzo często Ctrl–D).

`case` przełącza tryb tłumaczenia wielkich liter na małe podczas przesyłania plików komendą `mget` (patrz str. 7.6); normalnie zamienianie jest wyłączone, po włączeniu nazwy wszystkich plików, które są **całe** złożone z wielkich liter, będą zamieniane na małe; kolejne użycie komendy `case` wyłącza tą funkcję.

`cd `*`remote--directory`* ustawia kartotekę roboczą na zdalnym komputerze na *remote–directory*.

`cdup` zmienia kartotekę roboczą (na zdalnym komputerze) na wyższą od bieżącej.

`chmod `*`mode file--name`* zmienia prawa dostępu pliku o nazwie *file–name* na *mode*.[3]

`close` zamyka połączenie ze zdalnym komputerem (nie kończąc pracy programu **ftp**).

`delete `*`remote--file`* kasuje plik *remotefile* (na zdalnym komputerze); *remote–file* może być nazwą pustej kartoteki.[4]

`dir [`*`remote--directory`*`] [`*`local--file`*`]` listuje zawartość wskazanej kartoteki (lub bieżącej) na zdalnym komputerze, listing wypisywany jest na standardowym wyjściu lub do zbioru *local–file* gdy taką nazwę podano; gdy pracujemy w trybie interakcyjnym (patrz komenda `prompt` str. 7.6) **ftp** pyta czy drugi argument jest na prawdę nazwą pliku. w którym ma być zapisany listing.

`disconnect` synonim komendy `close` (str. 7.6)

`get `*`remote--file`*` [`*`local--file`*`]` kopiuje zbiór *temoet–file* (ze zdalnego komputera) do zbioru o nazwie *local–file*; jeżeli drugiego parametru nie podano zbiór kopiowany jest pod swoją oryginalną nazwą (może być zmieniona przez działanie komend `case` (str 7.6), `ntrans` (str 7.6) i `nmap` (str 7.6)).

`glob` przełącza „rozszerzanie” nazw plików zgodnie z konwencją stosowana w **csh**, (Niektóre znaki `*`, `?`, `[`, `]`, `~`, `{`, `}` mają specjalne znaczenie: `*` – oznacza zbiór o dowolnej nazwie, `a*` – oznacza zbiór o nazwie zaczynającej się na literę „a”, `[ab]*` – oznacza pliki o nazwach zaczyunających sie na „a” lub „b”, itd.).

[3] Użytkownik musi mieć do tego prawo.
[4] Użytkownik musi mieć do tego prawo.

hash przełącza drukowanie znaków # po przesłaniu każdego bloku (1024 bajtów) danych, pozwala to optycznie śledzić tempo przesyłania informacji.

help [*command*] wyświetla podstawowoe informacje o dostępnych komendach (gdy nie podano parametru *command* lub o wybranej komendzie).

idle [*seconds*] ustawia czas, po którym serwer zrywa połączenie, jeżeli klient nie jest aktywny, gdy nie podano parametru – wypisuje czas po którym połączenie będzie zerwane.

lcd [*local--directory*] zmienia lokalną kartotekę roboczą na kartotekę *local–directory*, kolejne pliki będą transmitowane do/z tej kartoteki, gdy parametr nie został podany – roboczą kartotęką staje się kartoteka główna („*home*") użytkownika.

ls [*remote--directory*] [*local--file*] odpowiednik komendy `dir` (str. 7.6) (na niektórych komputerach komenda `dir` daje wydruk szczegółowy, a komenda `ls` skrócony).

macdef *macro--name* definiuje makro (ciąg komend, które będą każdorazowo wykonane po podani nazwy *macro–name*). Kolejne linie (aż do linii pustej) będą zapisane jako treść makra. **ftp** narzuca ograniczenia: max 16 makr o łącznej długości 4096 znaków może być zdefiniowanych. znaki `$` i \ mają specjalne znaczenie. `$` po którym następuje liczba jest zastępowane w trakcjie „rozwijania" makra przez odpowiedni parametr występujący w wywołaniu makra. Znak \ przywraca oryginalne znaczenie znakowi `$`, to znaczy `\$` zabiera dolarowi specjalne znaczenie. ($i ??????????????)

mdelete [*remote--files*] kasuje zbiory na komputerze, z którym się połączyliśmy.[5] Jeżeli *globbing* (komenda `glob`, str. 7.6) jest włączony można kasować wiele zbiorów.

mget *remote--files* kopiuje zbiory ze zdalnego komputera. Nazwy plików poddawane są konwersjom wynikającym z zastosowanych komend `case` (str. 7.6), `ntrans` (str. 7.6) i `nmap` (str. 7.6).

mkdir *directory--name* teorzy na oddalonym komputerze kartotekę o nazwie *directory–name*[6]

mls *remote--files local--file* zapisuje skrócony listing kartoteki *remote-files* do pliku *local–file*

modtime *remote--file* podaje czas ostatniej modyfikacji pliku *remote–file*.

mput *local--files* kopiuje pliki z lokalnego komputera na zdalny; pliki są zapisywane pod takimi samymi nazwami jak na komputerze lokalnym (po zastosowaniu ewentualnych translacji wynikających z komend `ntrans` (str. 7.6) i/lub `nmap` (str. 7.6)); jeżeli włączony jest *globbing* (generacja nazw plików zgodnych z wzorcem) odpowiednie znaki są rozszerzane (patrz komenda `glob` (str. 7.6));

newer *file--name* sprowadza plik wtedy tylko gdy plik na komputerze zdalnym jest nowszy niż plik na komputerze lokalnym (gdy pliku na komputerze lokanlym nie ma – wykonywana jest komenda `get`)

nlist [*remote--directory*] [*local--file*] zapisuje listing wskazanej kartoteki do pliku, gdy kartotekanie została wyspecyfikowana – listowana jest bieżąca kartoteka robocza na komputerze zdalnym,

nmap [*inpatern outpatern*] przełącza mechanizm transformacji nazw plików. Jeżeli nie podano dodatkowych parametrów – mechanizm jest wyłączany. Gdy włączony jest mechanizm transformacji, poddawane są mu wszystkie nazwy podczas wykonywania komend `mget` (str. 7.6) i `mput` (str. 7.6) oraz komend `put` (str. 7.6) i `get` (str. 7.6), w których nie podano drugiego argumentu (nazwa pod jaką ma być zapisany plik).

Występujące we wzorcach ciągi znaków `$1`, `$2`, ... `$9`, mają specjalne znaczenie, \ używany jest do przywrócenia normalnego znaczenia znakowi `$`. Pozostałe znaki mają swoje normalne znaczenie.

Na przykład jeżeli *inpattern* ma postać `$1.$2;$3`, kopiowany plik ma nazwę `MYDATA.DATA;2` `$1` przyjmie wartość `MYDATA`, `$2` – `DATA` i `$3` – `2`.

outpattern określa, jak będzie wyglądała nazwa pliku po transformacji. `$1`, `$2`, ..., `$9` są zastępowane przez odpowiednią wartość z *inapattern*. Łańcuch `$0` zastępowana jest przez oryginalną nazwę zbioru. Dodatkowo ciąg znaków `[seq1,seq2]` jest zastępowana przez `seq1` jeżeli nie jest on pustym ciągiem znaków a przez `seq2` w przeciwnym przypadku.

[5] Użytkownik musi mieć do tego prawo.
[6] Użytkownik musi mieć do tego prawo.

W naszym przykładzie, jeżeli *outpattern* ma postać `$1.$2` zbiór `MYDATA.DATA;2` zostanie zapisany pod nazwą `MYDATA.DATA`,[7] ale można konstruować znacznie bardziej skomplikowane transformacje nazw plików

Komenda jest bardzo przydatna przy kopiowaniu wielu plików z komputera używającego innej konwencji nazewnictwa plików – na przykład z CD–ROMów[8] na komputerze `ldhpux.immt.pwr.wroc.pl`

`ntrans [`*`inchars`*` [`*`outchars`*`]]`

`open` *`server--host`* `[`*`prt-number`*`]` otwiera połączenie z komputerem *server–host* używając portu (bramy) o numerze *port–number* (jeżeli wyspecyfikowano).

`prompt` przełącza opcję interakcyjnego potwierdzania. Z domniemania **ftp** prosi o potwierdzenie przesłania każdego pliku w trakcie wykonania komend `mget` (str. 7.6), `mput` (str. 7.6) i dodatkowo podczas listowania zawartosci kartoteki do lokalnego pliku (patrz komenda `dir` na str. 7.6).

`proxy` komenda ta pozwala nawiązać połączenie ftp z dwoma komputerami i sterować przesyłaniem plików między dwoma komputerami z trzeciego. Radzimy wydać komendę `proxy ?` żeby sprawdzić jakie komedy są dopuszczalne.

`put` *`local--file`* `[`*`remote--file`*`]` kopiuje *local–file* na zdalny komputer (pod nazwą *remote–file* – jeżeli ją podano).

`pwd` wypisuje nazwę bieżącej kartoteki (na zdalnym komputerze).

`quit` synonim komndy `bye` (str. 7.6)

`quote` *`arguments`* wysyła *arguments* w niezmienionej postaci do serwera. W zależności od możliwości serwera można realizować różne ciekawe funkcje. Aby sprawdzić jakie możliwości daje serwer należy wydać komendę `quote help`. Szczególnie istotna może być realizowana przez serwer komenda `index` pozwalająca przeszukiwać strukturę plików zdalnego archiwum. `quote help index` podaje więcej szczegółów na temat tej komendy.

`recv` *`remote--file`* `[`*`local--file`*`]` synonim komendy `get` (str. 7.6).

`reget` *`remote--file`* `[`*`local--file`*`]` działa analogicznie jak komenda `get` (str. 7.6) z tą różnicą, że jeżeli *local–file* istnieje i jest krótszy niż *remote–file* to jest traktowany jako plik przesłany częściowo i przesyłanie pliku jest wznawiane od miejsca w którym zostało przerwane.[9]

`rhelp [`*`command--name`*`]` żąda informacji o komendach (gdy nie podano argumentu) realizowanych przez serwer lub o wyspecyfikowanej komendzie.

`rstatus [`*`file--name`*`]`

`rename` *`remote--from remote--to`*

`reset`

`restart` *`marker`* pozwala wznowić (przerwane przez utratę połączenia przesyłanie pliku). Działa podobnie jak komenda `reget` (str. 7.6), z tą tylko różnicą, że użytkownik sam musi sprawdzić długość przesłanego pliku, przekazać ją serwerowi za pomocą komendy `restart` i następnie wydać komendę `get` (str. 7.6); działnnie komendy `get` jest modyfikowane przez wcześniej wydaną komende `restart` tak, aby rozpocząć przesyłanie od zadanego „miejsca”.[10]

`rmdir` *`remote--directory`* kasuje kartotekę[11] na zdalnym komputerze (kartoteka musi być pusta!).

`runique` przełącznik umożliwiający włączanie/wyłączanie możliwości (normalnie wyłączona) generowania (przez program **ftp**) unikalnych nazw dla sprowadzanych plików w sytuacji, gdy istnieje już plik o nazwie identycznej z nazwą pliku sprowadzanego; unikalność nazw jest zapewniana przez dodawanie do nazwy pliku najpierw `.1`, `.2` do nazwy następnego (i tak, aż do `.99`) – gdy osiągnięta zostanie liczba 99 drukowany jest komunikat o błędzie i plik nie jest sprowadzany,

[7] Jeżeli dodatkowo użyjemy komendy `case` (str. 7.6) nazwa zostanie przetransformowana do `mydata.data`.
[8] http://www.immt.pwr.wroc.pl/htbin/cdrom
[9] Nie wszystkie serwery realizują taką funkcję!
[10] Nie wszystkie serwery realizują taką funkcję!
[11] Użytkownik musi mieć do tego prawo.

`send` *`local--file`* `[`*`remote--file`*`]` synonim komendy `put` (str. 7.6).

`site` ***`arguments`*** synonim komendy `quote` (str. 7.6) ?????????????????.

`size` ***`remote--file`*** zwraca wielkość pliku (na zdalnym komputerze).

`status` informuje w jakim stanie znajduje się program **ftp**.

`sunique` „przełącznik" umożliwiający włączenie/wyłączenie możliwości generowania przez serwer unikalnych nazw plików w sytuacji gdy przesyłany plik miał by skasować plik o o tej samej nazwie co plik przesyłany,

`system` zwraca informacje o typie systemu operacyjnego pracującego na zdalnym komputerze.

`tenex` ustawia tryb transmisji plików na *tenex*.

`type [`*`type--name`*`]`

`user` *`user--name`* `[`*`password`*`] [`*`account`*`]`

`verbose`

`?` – synonim komendy `help` (str. 7.6).

7.7 Windows 95

Rozdział ten jest jednym z kilku (inne opisują programy **telnet** (str. 6.1.1), **tracert** (str. 4.4.1) i **ping** (str. 4.3.1)) opisujących podstawowe usługi sieciowe dostępne w systemie Windows 95.

ftp w wykonaniu firmy Microsoft nie może być uznane za produkt godny zainteresowania. Jedyną jego zaletą jest to, że jest. Z drugiej strony jest to bardzo klasyczna implementacja **tekstowej** wersji tego programu, o możliwościach zbliżonych do innych tego typu programów w środowiskach tekstowych (MS DOS, UNIX).

Program uruchamiamy w sposób następujący:
`ftp [-v] [-n] [-i] [-d] [-g] [`*`host`*`] [-s:` *`filename`*`]` Znaczenie parametrów jest następujące:

`-v` wyłącza wyświetlanie informacji generowanych przez serwer, z którym się łączymy,

`-n` wyłącza możliwość automatycznego logowania się na serwer,

`-i` wyłącza tryb potwierdzania przesyłania każdego pliku podczas wykonywania poleceń `mget` i `mput`,

`-d` włącza tryb śledzenia (wyświetlane będą dodatkowe, bardziej szczegółowe informacje na temat przebiegu połączenia i przesyłania informacji),

`-g` wyłącza tryb rozpoznawania znaków specjalnych (funkcja identyczna do tej w implementacji programu **ftp** w systemie UNIX),

host adres komputera, z którym się łączymy,

`-s:` ***filename*** określa nazwę pliku, z którego pobierane będą polecenia wykonywane zaraz po nawiązaniu połączenia z serwerem.

Podczas pracy z programem **ftp** wydawać można polecenia, które są zasadniczo zgodne z tymi znanymi z innych implementacji. Poniżej podamy jedynie listę dostępnych poleceń i znaczenie jedynie tych, które nie są dostępne (lub funkcjonują zasadniczo inaczej):
`!`, `?`, `append`, `ascii`, `bell`, `binary`, `bye`, `cd`, `close`, `debug` (przełącza tryb wyświetlania komunikatów diagnostycznych), `delete`, `dir`, `disconnect`, `get`, `glob`, `hash`, `help`, `lcd`, `literal` (przesyła pojedyńcze polecenie do serwera ftp, odpowiednik polecenia `quote`), `ls`, `mdelete`, `mdir` (pozwala uzyskać listing wielu plików lub kartotek, przykładowe użycie `mdir a*`), `mget`, `mkdir`, `mls` (odpowiednik polecenia `mdir`, informacje wyświetlane są w postaci skróconej, podobnie jak w przpadku polecenia `ls`), `mput`, `open`, `prompt`, `put`, `pwd`, `quit`, `quote`, `recv`, `remotehelp` (wyświetla informacje o wszystkich poleceniach rozpoznawanych przez serwer ftp z którym jesteśmy połączeni), `rename`, `rmdir`, `send`, `status`, `trace` (wyświetla informacje o każdym pakiecie podczas pracy programu **ftp**), `type`, `user`, `verbose` włącza (lub wyłącza) tryb szczegółowego informowania o wszystkich komunikatach podczas trwania połączenia z serwerem ftp.

7.8 MS DOS i PC/TCP

Pełna postać komendy uruchamiającej program **ftp** jest następująca:

`ftp [-d][-u` *userid password*`][-p` *port_no*`]` *host* `[`*command*`]`
`ftp [-?|-version]`

Parametry umieszczane w linii komendy:

`-d` program podczas pracy wyświetla szczegóły „konwersacji" z serwerem;

`-u` ***userid password*** pozwala na automatyczne połączenie z serwerem używając jako nazwy użytkownika *userid*, a jako hasła *password*;

`-p` ***port_no*** specyfikuje numer portu, z którym łączy się klient;

host podaje nazwę (lub adres IP) komputera, z którym chcemy się połączyć;

command wykonuje podaną komendę (dopuszczalne komendy wyszczególnione są poniżej);

`-?` wyświetla podstawowe informacje o parametrach komendy **ftp**;

`-version` podaje numer wersji programu (ważne w przypadku zgłaszania błędów producentowi oprogramowania).

Opis komend używanych podczas pracy z programem **ftp**.

`! [`***command***`]` gdy nie podano argumentu – wyjście do DOSa (powrót do pracy z **ftp** następuje po napisaniu komendy `exit`); gdy podano argument – powoduje wykonanie podanej komendy i powrót do pracy z **ftp**;

`? [`***command***`]` gdy nie podano argumentu – drukuje wszystkie, rozpoznawane przez **ftp** komendy; gdy podano nazwę komendy – wypisuje krótki jej opis;

`acct [`***account_name***`]` (większość serwerów nie realizuje tej funkcji) w zamierzeniu komenda pozwala w różny sposób rozliczać wykonywane przez użytkownika operacje;

`ascii` przełącza w tryb tekstowy przesyłania danych;

`binary` zezwala na przesyłanie zbiorów binarnych pomiędzy podobnymi komputerami; komenda `local` *n* pozwala na poprawne przesyłanie zbiorów binarnych pomiędzy różnymi typami maszyn;

`bye` synonim komendy `exit` (str. 7.8);

`cd` zmienia bieżącą kartotekę na zdalnym komputerze (serwerze);

`debug on|off` włącza (`on`) lub wyłącza (`off`) tryb drukowania komunikatów diagnostycznych, nie jest normalnie wykorzystywana;

`delete` ***filename*** kasuje wskazany zbiór na serwerze (użytkownik musi mieć uprawnienia do kasowania zbiorów);

`dir [`***argument*** `[`***filename***`]]` listuje zawartość bieżącej kartoteki serwera; gdy użyty jest pierwszy argument komendy (*argument*) może on wskazywać dokładna nazwę kartoteki, podkartoteki, zbioru lub być wzorcem (*wildcard*) ograniczającym listowane zbiory. Gdy wyspecyfikujemy *filename* listing zawartości kartoteki zostanie zapisany w zbiorze o podanej nazwie;

`drive` ***driveletter*** zmienia bieżący (domniemany) dysk na komputerze PC;

`exit` przerywa połączenie z serwerem i kończy pracę **ftp**;

`fcd` ***path*** synonim komendy `cd` (str. 7.8); zdarzają sie serwery nie realizujące tej komendy;

`fpwd` drukuje aktualną kartotekę serwera (nie wszystkie serwery realizują tę komendę);

`get [`***foreign_file*** `[`***local_file***`]]` kopiuje zbiór z serwera na lokalny komputer PC (w aktualnie zadeklarowanym trybie); zbiór zapisywany jest w bieżącej kartotece pod swoją nazwą (lub, jeżeli podano drugi argument – pod nową nazwą); jeżeli pominięto obie nazwy użytkownik zostanie odpytany o nazwę zbioru, który ma być kopiowany i nazwę pod jaką ma być zapisany;

`iget` [*foreign_file* [*local_file*]] podobnie jak `get`, ale kopiuje zbiór w trybie `image`, po skopiowaniu powraca do poprzedniego trybu;

`image` synonim komendy `binary`;

`iput` [*local_file* [*foreign_file*]] podobnie jak `put`, ale zbiór przesyłany jest w trybie binarnym;

`lcd` [*local_directory*] zmienia bieżącą kartotekę na komputerze PC;

`ldir` [*path filename*] listuje zawartość lokalnej kartoteki;

`local` *n* pozwala na binarne przesyłanie zbiorów z/na komputer o innej długości bajtu niż na komputerze PC;

`login` *user* ustala nazwę użytkownika na zdalnym komputerze; komenda jest przydatna w przypadku pomyłki przy podawaniu hasła (nawiązane jest logiczne połączenie pomiędzy komputerami!);

`lpwd` drukuje nazwę aktualnej kartoteki na komputerze PC;

`ls` [*argument* [*filename*]] działanie podobne jak komendy `dir`, ale listing jest „krótki” (niektóre serwery nie rozróżniają komend `dir` i `ls`, dając w każdym przypadku „długi” listing, zawierający oprócz nazw zbiorów ich długość i prawa dostępu);

`mdelete` *pattern* kasuje wiele zbiorów (o nazwie zgodnej ze wzorcem) na zdalnym komputerze; jeżeli ustawiona jest opcja `opton ask on`, program pisze nazwe zbioru i pyta się czy go skasować; poprawnymi odpowiedziami są:

- `n` (no) nie kasuj zbioru,
- `p` (proceed) kasuj ten i wszystkie następne zbiory bez pytania,
- `q` (quit) zatrzymaj się natychmiast,
- `y` (yes) tak, skasuj zbiór.

`mget` *pattern* powoduje przesłanie wielu zbiorów (o nazwie zgodnej ze wzorcem); gdy ustawiona jest opcja `option ask on` program wypisuje nazwy kolejnych zbiorów i pyta czy przesłać; poprawne odpowiedzi:

- `a` (ASCII) przesyłaj (od tej chwili) w trybie tekstowym,
- `i` (image) przesyłaj (od tej chwili) w trybie binarnym,
- `n` (no) nie przesyłaj tego zbioru,
- `p` (proceed) prześlij ten zbiór i wszystkie następne bez pytania,
- `q` (quit) zatrzymaj się natychmiast,
- `r` (rename) zapytaj o nową nazwę zbioru,
- `t` (tenex) przesyłaj (od tej chwili) wszystkie zbiory w trybie `tenex` (`local byte size 8`),
- `y` (yes) prześlij ten zbiór.

`mkdir` *foreign_directory* tworzy nową kartotekę na zdalnym komputerze (nie wszystkie serwery realizują tę funkcję); użytkownik musi mieć prawo tworzenia kartotek;

`mput` *pattern* komenda powoduje przesłanie wielu zbiorów (o nazwach zgodnych z wzorcem) na zdalny komputer; jeżeli ustawiona jest opcja `option ask on` program podaje nazwę zbioru i pyta czy ma być przesłany; można odpowiadać tak samo jak w przypadku komendy `mget`;

`option` [*option_name*] `on|off` gdy nie wyspecyfikowano żadnych argumentów, komenda powoduje wypisanie aktualnych ustawień parametrów; opcje mogą być następujące:

- `ask` (normalnie `off`) gdy jest on, ftp odpytuje czy przesyłać/kasować zbiory (komendy `mget`, `mput` i `mdelete`),
- `casehack` (normalnie `on`) w trybie `casehack on` podczas automatycznego tworzenia nazwy zbioru na komputerze docelowym, nazwa zbioru pisana jest małymi literami; w trybie `off` dużymi; może to mieć istotne znaczenie podczas przesyłania zbiorów na komputer który rozróżnia małe i wielkie litery w nazwach zbiorów,
- `hash` (normalnie `off`) drukuje znak # po każdym przesłanym bloku danych,

`page` (normalnie `off`) steruje funkcją stronicowania podczas długich listingów (komendy `dir i show`); specjalna postać komendy `option page` *n* pozwala określić wielkość strony (normalnie 23 linie),

`pathhack` (normalnie `on`) podczas automatycznego generowania nazwy zbioru docelowego **ftp** stara się odrzucić z niej nazwę kartoteki; po wyłączeniu tej opcji można specyfikować pełną ścieżkę.

`parent` powoduje zmianę bieżącej kartoteki na kartotekę główną (nie wszystkie serwery realizują tę funkcję);

`passive` wykonuje następne przesłanie danych w trybie pasywnym (połączenie do przesłania danych inicjuje PC, a nie serwer, jak normalnie); nie wszystkie serwery realizują tę funkcję;

`put [`***local_file remote_file***`]` kopiuje *local_file* do *remote_file* w aktualnie obowiązującym trybie;

`pwd` synonim komendy `fpwd`;

`quit` synonim komendy `exit`;

`quote` ***command*** wysyła komendę do wykonania bezpośrednio do serwera; `quote help` wypisuje listę możliwych do użycia komend;

`rename [`***old_name new_name***`]` zmienia nazwę zbioru na zdalnym komputerze;

`retrieve [`***remote_file old_file***`]` synonim komendy `get`;

`rmdir` ***foreign_directory*** kasuje kartotekę na komputerze, z którym jesteśmy połączeni; kartoteka musi być pusta, użytkownik musi mieć prawo jej skasowania;

`send [`***local_file remote_file***`]` synonim komendy `put`;

`server` ***command*** synonim komendy `quote`;

`show` ***filename*** wyświetla zawartość wskazanego zbioru na ekranie;

`stat` pyta serwer o jego aktualny stan (nie wszystkie serwery realizują tę komendę);

`store [`***local_file remote_file***`]` synonim komendy `put`;

`take` ***local_file*** czyta i wykonuje komendy z lokalnego zbioru, pozwala to automatyzować przesyłanie dużych zbiorów; w zbiorze komend mogą znajdować się wszystkie komendy rozpoznawane przez **ftp**; linie zaczynające się od znaku # traktowane są jako komentarz;

`tenex` pozwala dokonywać poprawnych przesłań zbiorów binarnych z komputerów TOPS-20;

`tget [`***foreign_file*** `[`***local_file***`]]` podobnie jak komenda `get`, ale przesyłanie odbywa się w trybie `tenex`;

`tput [`***local_file***`[`***foreign_file***`]]` podobnie jak `put`, ale przesyłanie odbywa się w trybie `tenex`;

`type` ***type*** gdy nie podano argumentu, komenda powoduje wyświetlenie aktualnego trybu przesyłania informacji; jeżeli został wyspecyfikowany argument – powoduje ona ustalenie trybu przesyłania informacji; *type* może przyjmować jedną z następujących wartości: `image`, `ascii`, `tenex`, `local` *n*;

`user` ***name*** synonim komendy `login`;

`version` wyświetla numer wersji programu **ftp**;

Rozdział 8

Drukowanie plików

Z najprostszą sytuacją mamy do czynienia wtedy, gdy drukarka podłączona jest bezpośrednio do komputera, ale tym nie będziemy się zajmować.

Zajmiemy się tu problemem zcentralizowanego drukowania na drukarce podłączonej do sieci lub obsługiwanej centralnie przez jeden komputer.

W komputerach UNIXowych spotkać możemy dwa, niezbyt ze sobą zgodne systemy obsługi drukarek:

- wywodzący się z systemu „BSD" (drukarka obsługiwana jest zazwyczaj komendą **lpr**),
- wywodzący sie z systemu V (drukarka obsługiwana jest komendą **lp**).

Systemem pomyślanym od początku jako „sieciowy" i „rozproszony" jest system wywodzący się z BSD. Oczywiście wraz z upływem czasu różnice zacierały się i rozmazywały, co uniemożliwia praktycznie jednolite opisanie komend drukowania. Co gorsze zestaw dostępnych możliwości zależy od rodzaju drukarki i możliwości oprogramowania.

Na przykład system drukowania w oprogramowaniu MPower systemu HP–UX pozwala wysyłać za pomocą programu **lp** pliki o różnej zawratości (PostScriptowe, pliki .gif, tiff, jpg, ...) W zależności od ich zawartości uruchamiany jest odpowiedni program konwersji i plik jest poprawnie drukowany.

Opiszemy tu program **lpr** z pakietu PC/TCP, ktory jakoś współpracuje z drukarką sieciową w Laboratorium Dynamiki. Opis ten można w pierwszym przybliżeniu zastosować do programu **lpr** dostępnego w pakiecie oprogramowania PD rozpowszechnianym pod nazwą **NCSA telnet**.

8.1 lpr

Program **lpr** komunikuje się z serwerem zarządzającym drukowaniem na komputerze UNIXowym i przekazuje mu zbiory do wydrukowanaia. Serwer, z reguły, ma określone nazwy komputerów które mogą przekazywać mu zlecenia drukowania i przyjmuje je na ogół jedynie z lokalnej sieci. Możliwe jest jednak takie skonfigurowanie całego systemu, aby nasze zbiory były drukowane w oddalonym punkcie (tylko po co?).

Użycie:

`lpr [-S` *`server`*`] [-P` *`printer`*`] [-C` *`class`*`] [-J` *`jobname`*`] [-#` *`number`*`] [-hpltndgvcf] [`*`filename...`*`]`
`lpr -?|-version`

Znaczenie poszczególnych parametrów jest następujące:

`-S` *`server`* nazwa lub adres IP komputera, na którym uruchomiony jest serwer, z którym kontaktujemy się, (normalnie nie musi być podawane; oprogramowanie jest tak skonfigurowane, że prawidłowo przesyła wszystkie wydruki do jedynego serwera w naszej sieci lokalnej),

`-P` *`printer`* nazwa drukarki, na której chcemy drukować, w naszym przypadku jedna z nazw: laser, deskjet, robotron, epson określających jednoznacznie drukarkę,

`-C` *`class`* w instalacjach, gdzie jest kilka drukarak tego samego typu można utworzyć „klasę" grupującą je; użytkownik może drukować na dowolnej z nich podając tylko nazwę klasy; wybrana zostanie najmniej obciążona drukarka,

`-J` *`jobname`* określa nazwę wydruku, jeżeli nie poda się tego parametru system automatycznie wygeneruje nazwę zadania na podstawie nazwy drukowanego zbioru,

-# *number* określ liczbę kopi zbioru, która będzie drukowana,

`-h` nie drukuje strony nagłówka (dla oszczędności papieru system został tak skonfigurowany, że strony nagłówka nie są drukowane),

`-p` poleca serwerowi, aby tekst wydruku sformatował wykorzystując program **pr**,[1]

filename nazwazbioru/zbiorów, które mają być drukowane, gdy zostanie pominięta **lpr** czyta dane z wejścia standardowego i może być użyty w konstrukcjach postaci: `dir | lpr`,

Program **lpr** jest dosowską implementacją **Unix**owego systemu drukowania. Wykorzystywane jest tam pojęcie *filtru* czyli programu dokonującego przetwarzania przesłanego do serwera pliku.

W Laboratorium Dynamiki nie został zaimplementowany żaden z tych filtrów. Niemniej jednak podajemy wszystkie opcje programu **lpr** mające zastosowanie w tym przypadku. Dane binarne powinny być drukowane z użyciem opcji `-v`.

`-l` poleca serwerowi, aby drukowane były wszystkie znaki sterujące zawarte w tekście,

`-t` informuje server, że drukowane dane pochodzą z programu formatującego **troff**,

`-n` informuje server, że drukowane dane pochodzą z programu formatującego **ditroff**,

`-d` informuje server, że przesyłany jest zbiór `.dvi` (przetworzony TEXem zbiór .tex); przetwarzanie z postaci `.dvi` do postaci drukowalnej wykonywane jest na zlecenie serwera; **uwaga: opcja ta nie jest zaimplementowana w naszej sieci!!**

`-g` informuje serwer, że przesyłany jest zbiór wyjściowy utworzony przez procedury **plot**; przetwarzanie do postaci drukowalnej następuje na zlecenie serwera,

`-v` nie dokonywana będzie żadna translacja przesyłanych danych; opcja powinna być wykorzystywana przy drukowaniu danych binarnych,

`-c` informuje serwer, że dane pochodzą z programu **cifplot**,

`-f` poleca serwerowi, aby przesłane dane interpretował tak jak dane wyjściowe pochodzące z progarmu napisanego w Fortranie, informacja sterująca wydrukiem zawarta jest w pierwszym znaku każdego rekordu,

8.2 predir, onpredir, dopredir

Aby dokonać przechwycenia danych wysyłanych do drukarki przez programy pracujące pod kontrolą systemu operacyjnego MS DOS musi być zastosowany program rezydentny **predir**. Jest on stosunkowo mały (zajmuje ok 7 kB) i aby móc zrealizować wszystkie swoje funkcje współpracuje z dwoma programami **onpredir** i **dopredir**. Nie wykonują one żadnych istotnych czynności poza interpretacją poleceń użytkownika i przekazywaniem ich programowi **predir**.

Użycie:

```
predir [config] [debug] [help] [hotkey opt] [memsize xx] [noselect opt]
       [oneof opt] [onexit opt] [timeout opt] [printer port] [prog program]
       [report] [spool dest] [swap dest] [unload]
predir -?|-version
```

Znaczenie parametrów jest następujące:

`config` ładuje standardową konfiguracje zapisaną w zbiorze konfiguracyjnym,

`debug` wyświetla dodatkowe informacje na temat parametrów pracy programu,

`help` wyświetla podstawowe informacje o programie,

`hotkey` *opt* włącza/wyłącza rozpoczęcie drukowanie po naciśnięciu klawiszy Ctrl–LeftShift–RightShift; *opt* przyjmuje wartości `on` lub `off`,

`memsize` *xx* określa wielkość obszaru pamięci (dyskowej, w kilobajtach) używanej przez program do przechowywania wydruków na dysku; $63 < xx < 129$,

[1] Program ten drukuje standardowe nagłówki na każdej stronie, numeruje strony, itp.

`noselect` *`opt`* zezwala lub zabrania ignorować zawartość przekazywaną w rejestrze DX podczas wywoływania Int17,

`oneof` *`opt`* włącza i wyłącza rozpoczęcie drukowania po wykryciu znaku EOF (end of file),

`onexit` *`opt`* włącza/wyłącza rozpoczęcie drukowania po zakończeniu programu użytkowego,

`printer` *`port`* określa drukarkę, nad którą **predir** przejmuje kontrolę; dopuszczalne wartości: PRN, LPT1, LPT2, LPT3, LPT4 lub cyfra od 0 do 3 (0=LPT1, 3=LPT4); wyłączenie drukowania następuje gdy *port* przyjmuje wartość `off`,

`prog` *`program`* określa nazwę programu lub zbioru `.bat` wykonującego funkcję przesłania zbioru na zdalny komputer i wydrukowania go; nazwa powinna być pełną nazwą pliku,

`report` program wypisuje aktualną konfigurację,

`spool` *`dest`* określa kartotekę i/lub nazwę zbioru tymczasowego używanego do przechowywania „przechwyconego" wydruku (normalnie nie musi być używane),

`swap` *`dest`* określa kartotekę i/lub nazwę zbioru używanego do zapisu zawartości części pamięci operacyjnej na czas drukowania (normalnie nie musi być używane),

`timeout` *`opt`* ustawia opcję rozpoczęcia automatycznego drukowania, gdy program użytkowy przez określony czas (wyrażony w sekundach) nie wysyła żadnych informacji do drukarki; aby możliwość tę wyłączyć należy ustawić czas na 0,

`unload` usuwa **predir** z pamięci, jeżeli jednak po załadowaniu go do pamięci załadowano inne programy rezydentne – usunięcie jest niemożliwe.

Działanie programu **predir** jest następujące: po uruchomieniu go, ładuje się rezydentnie do pamięci, przechwytując przerwanie Int17 (służące do komunikowania się programów użytkowych z drukarką). Następnie przechwytuje wszystkie odwołania do drukarki, zapisując dane (które normalnie przesyłane są do drukarki) w zbiorze dyskowym. Zawartość zbioru jest następnie drukowana gdy nastąpi jeden z warunków:

- użytkownik zażąda wydruku naciskając specjalną sekwencję klawiszy (`hotkey on`) lub uruchomi program **dopredir**,
- program użytkowy wyśle znak końca zbioru do drukarki (`oneof on`),
- program użytkowy zakończy pracę (`onexit on`),
- program użytkowy przez określony czas nie wysyła informacji do drukarki (`timeout` *`opt`*),

W najprostszej wersji drukowanie wykonywane jest z wykorzystaniem programu **lpr** (na domniemanej drukarce). Można utworzyć specjalne zbiory komend pozwalające zmieniać jego zachowanie.

Na czas wydruku (gdy trzeba uruchomić program **lpr**) **predir** musi zwolnić część pamięci operacyjnej – wykorzystywany jest zbiór pośredni na dysku; uruchomienie programu **lpr** nastąpić może „równocześnie" z pracą programu użytkowego, i jest to potencjalnie niebezpieczne.

Po uruchomieniu programu **predir** jego zachowanie można zmienić za pomocą programu **onpredir** (używającego dokładnie takich samych jak dla **predir** parametrów).

Rozdział 9

Poczta elektroniczna

9.1 Poczta SMTP

Najpopularniejszą usługą sieciową jest niewątpliwie poczta elektroniczna (zwana również czasami z angielska „*e–mail*"). Pozwala ona na wymianę informacji pomiędzy pojedyńczymi osobami, a dzięki instytucji listy dyskusyjnej pomiędzy grupami osób.

Tak jak w każdej formie komunikacji i tu jej uczestnicy pragną porozumiewać się w swoim języku wykorzystując wszystkie jego możliwości. Niestety, w standardowej poczcie elektronicznej (oznaczanej czasmi jako SMTP) jest to **niemożliwe.**

Jaka jest tego przyczyna?

Poczta elektroniczna powstała ona już bardzo dawno temu, co więcej w Stanach Zjednoczonych i nie uwzględniała w żaden sposób faktu istnienia znaków narodowych. (Można co prawda powiedzieć, że w tamtych czasach kodowano znaki narodowe zgodnie z normą ISO 646 która przeznaczała na ten cel znaki [,], {, }...)

Ten standard przesyłania informacji opisuje norma RFC 822[1]. A producenci oprogramowania zaimplementowali ją czasami tak rygorystycznie, że listy zawierające znaki "narodowe" o kodach powyżej 128 były albo odrzucane, albo zniekształcane.

Komputer PC przyniósł nowy standard i nowe zasady kodowania znaków. Okazało się jednak, że na w jednym bajcie (dzięki ośmiu bitom można zapisać 256 kombinacji) nie ma dość miejsca na uwzględnienie znaków używanych przez różne nacje. Kolejna norma (ISO 8859) składała się z dziewięciu części i definiowała dziewięć różnych alfabetów (tu wymieniam jedynie niektóre):

ISO–8859–1 zawiera znaki używane przez kraje Europy Zachodniej,

ISO–8859–2 zawiera znaki używane przez kraje Europy Wschodniej (kraje „słowiańskie" korzystające z alfabetu łacińskiego),

ISO–8859–5 cyrylica,

ISO–8859–6 znaki arabskie,

ISO–8859–7 greka,

ISO–8859–8 hebrajski.

(Żaden z nich bodaj nie rozwiązuje problemu języków dalekowschodnich.)

I w tym momencie rozpoczęły się problemy. List nadany w jednym standardzie mógł być odbierany przez użytkownika korzystającego z zupełnie innego zestawu znaków. Wymyślono standard MIME mający być lekarstwem na wszystkie bolączki. Został on stworzony przy założeniu, że:

1. Wszystkie wiadomości przesyłane są z wykorzystaniem znaków jednego z alfabetów wymienionych przez normę.
2. W nagłówku listu umieszczona jest informacja o użytym zestawie znaków.
3. Informacje mogą być przesyłane w postaci binarnej (zawierającej znaki z rozszerzonego zestawu ASCII) lub (co jest z różnych względów bardziej praktyczne[2]) w postaci zakodowanej. Sposób kodowania został tak opracowany, żeby przesyłka była w najmniejszym stopniu narażona na zniekształcenia.

[1] ftp://ftp.pwr.wroc.pl/pub/doc/rfc/rfc.0800-0899/rfc822.txt.gz

[2] W sytuacji gdy w dosyć powszechnym użyciu ciągle jest oprogramowanie pocztowe nie akceptujące znaków o kodach większych niż 128.

4. Program pocztowy odbiorcy powinien przesyłkę rozkodować i wyświetlić **najlepiej jak to jest możliwe** w środowisku odbiorcy. Oznacza to że być może powinien on dokonać przekodowania przesyłki ze standardu ISO na standard lokalny, używany na komputerze odbiorcy. W trakcie przekodowania, może okazać się, że znaki narodowe zostaną zastąpione przez najbliższe ich odpowiedniki dostępnego alfabetu: zamiast **ł** na ekranie zobaczymy **l**; a zamiast **ź** i **ż** bedziemy zawsze widzieć **z**.

Niestety, ciągle nie ma zbyt wielu programów pocztowych realizujących w pełni tę ideę. Co najgorsze, oprogramowanie pracujące w środowisku MS Windows (zwłaszcza to „darmowe") pisane jest głównie przez programistów amerykańskich (i nie zawsze uwzględnia potrzeby innych nacji).

A już najgorsze jest to, że standard kodowania polskich liter przyjęty w bardzo popularnym środowisku MS Windows (tak zwana strona kodowa CP 1250) różni się (dotyczy to zaledwie kilku liter) od standardu iso-8859-2.

Trzeba jednak powiedzieć, że najnowsze produkty firmy Microsoft: MS Internet News, MS Internet Mail, Microsoft Mail, Internet Explorer znakomicie radzą sobie z tym problemem.

Używanie „złego" programu pocztowego prowadzi do sytuacji w której:

1. Nadawca poczty (mając możliwość korzystania z polskich liter) pisze po polsku.
2. Źle skonfigurowany (lub nie dający się skonfigurować) program pocztowy wysyła taki list bez żadnego przekształacenia do postaci zgodnej ze standardem (bardzo często umieszczając w nagłówku **złą** informacje o sposobie kodowania).
3. Odbiorca (korzystający z „przypadkowego") programu pocztowego (lub źle skonfigurowanego, albo też nie dającego się skonfigurować) widzi na ekranie tak zwane „krzaczki" albo zamiast treści czytelnej (za wyjątkiem być może pojedyńczych znaków) widzi coś takiego:

```
From:          "Wojciech Myszka" <myszka>
To:            myszka@ldhpux.immt.pwr.wroc.pl
Date:          Sun, 18 Feb 1996 22:25:57 +0100
MIME-Version:  1.0
Subject:       To jest tez list zwierajacy polskie znaki
X-mailer: Pegasus Mail v3.30
Content-type: text/plain; charset=iso-8859-2
Content-transfer-encoding: BASE64
```


```
T2RiaW9yY2EgKGtvcnp5c3RhaqVjeSB6ICwscHJ6eXBhZGtvd2VnbycnKSBwcm9ncmFtdSBw
b2N6dG93ZWdvIChsdWIgq2xlCnNrb25maWd1cm93YW5lZ28sIGFsYm8gdGW+IG5pZSBkYWql
Y2VnbyBzaakgc2tvbmZpZ3Vyb3dhhikgd2lkemkgbmEKZWtyYW5pZSB0YWsgendhbmUgLCxr
cnphY3praScnIGFsYm8gemFtaWFzdCB0cmWYY2kgY3p5dGVsbmVqICh6YSB3eWqldGtpZW0K
YnmGIG1vvmUgcG9qZWR55GN6eWNoIHpuYWuidykgd2lkemkgY2+YIHRha2llZ286Cg==
```


Zniechęca to dosyć skutecznie do korzystania z polskich znaków w poczcie elektronicznej.

Zajmimy się zatem standardem MIME.

9.2 MIME

Jak już pisaliiśmy wcześniej poczta elektroniczna (*e–mail*) służy do przesyłania (krótkich, na ogół) wiadomości od jednego użytkownika sieci do drugiego. Narzędzie to, opracowane bardzo dawno (i w USA) ma szereg ograniczeń:

- służyć może jedynie do przesyłania tekstów ASCII (znakowych); co więcej, w czasach gdy definiowano standard [17], znaki ASCII były to znaki o kodach mniejszych od 128 (7 bitowe) i nie uwzględniały znaków „narodowych",
- mimo, że praktycznie wszystkie wersje współczesnych systemów operacyjnych są już ośmiobitowe (pozwalają używać w tekstach kodów tak zwanego „rozszerzonego zestawu ASCII") i posiadają odpowiednio skonfigurowane programy pocztowe pozwalające odbierać i nadawać pocztę ośmiobitową [10] – standard pozostał taki jaki był (siedmiobitowy) i nie można mieć pewności, że każdy komputer (nawet pośredni) pocztę ośmiobitową „przepuści",

- praktycznie każdy system, operacyjny używa innego sposobu kodowania znaków narodowych (Unix – ISO–8859–2, patrz zebrane o tym informacje od głównych producentów, DOS – CP852, Windows – CP1250 i własne standardy MACa i Amigi...) MIME zapewnia możliwość standardowego przesyłania informacji,
- nie nadaje sie do przesyłania wiadomości multimedialnych (zawierających na przykład głos lub obraz).

Problem jest oczywiście znany od lat i opracowano kilka metod jego rozwiązania. Jedną z nich jest MIME [2] czyli *Multi–purpose Internet Mail Extensions*, publicznie dostępny standard pozwalający na przesyłanie tekstów (również zawierających znaki narodowe), plików binarnych i wiadomości multimedialnych pomiędzy komputerami korzystającymi z niego.

Standard MIME pozwala na przesyłanie następujących wiadomości:

- zawierających znaki narodowe,
- zawierających „tekst wzbogacony" (*enriched tex*), czyli tekst zawierający dodatkowe informacje formatujące pozwalające na wytłuszczanie, centrowanie, itd.. . .
- obrazów,
- dźwięków,
- inne rodzaje wiadomości (odpowiednio opakowane),
- pliki binarne (na przykład „ztarowane"),
- pliki PostScriptowe,
- odsyłacze do plików zawartych w archiwach *anonymous ftp*,[3]
- innych rzeczy.

Standard daje też „standardową" metodę definiowania nowych typów przesyłania wiadomości (trzeba tylko pamiętać, żeby wytłumaczyć odbiorcy jak je przetwarzać – odtwarzać).

Oczywiście, od pewnego czasu, różni dostawcy oprogramowania dostarczają różne pakiety pozwalające na przesyłanie wiadomości multimedialnych. Niestety pakiety nie zawsze są „kompatybilne" ze sobą. MIME ma za zadanie rozwiązać ten problem. System został zaprojektowany bardzo starannie, tak żeby był zgodny z różnymi protokołami przesyłania poczty (SMTP,[4] UUCP,[5] MMDF[6] czy BITNET).

Nie każdy z opisanych (w planie :-) programów pocztowych daje jednakowe możliwości nadawania/odbierania przesyłek multimedialnych. . .

Pewną konkurencję, być może, będzie stanowił standard poczty X.400.

Jako dodatkową lekturę polecam tekst Piotra Piątkowskiego na Polskiej Stronie Ogonkowej.

9.3 Cechy charakterystyczne przesyłki MIME

Każda przesyłka pocztowa składa się z dwu zasadniczych części: nagłówków (coś na kształt „koperty") i treści przesyłki. Zawartość nagłówków szczegółowo określają następujące dokumenty: RFC822 i RFC1521. Nagłówki oddzielone są od treści listu jedną linią pustą i określają:

From od kogo otrzymano list (zwracam uwage: po from **nie** ma dwukropka);

From: adres nadawcy (tu **jest** dwukropek);

Received: punkty pośrednie (adresy węzłów przez które przesyła przeszła zanim trafiła do celu);

To: adres dobiorcy listu;

Subject: temat listu;

Date: data nadania przesyłki;

[3] Zamiast przesyłać plik przesyłamy tylko informacje skąd go ściągnąć – podczas odczytywania poczty, jeżeli zdecydujemy się, zostanie on automatycznie ściągnięty.

[4] Simple Mail Transport Protocol

[5] Unix to Unix CoPy

[6] Multichannel Mail Delivery Facility

Message-ID: numer identyfikacyjny przesyłki (każda przesyłka powinna mieć unikalny numer identyfikacyjny);

Reply-To: adres, na który należy przesłać odpowiedź na list;

Cc: adres, na który została wysłana kopia listu;

oraz pewną ilość nagłówków pomocniczych – informacyjnych o mniejszym na ogół znaczeniu.

Przesyłki MIME zawierają dodatkowo nagłówek:

Mime-Version: 1.0 który identyfikuje wszystkie takie przesyłki

oraz nagłówki określające:

Content-Type: rodzaj zawartości przesyłki,

Content-Transfer-Encoding: sposób jej kodowania,

Content-Length: długość przesyłki (nie zawsze występuje),

Content-description: opis zawartości przesyłki.

Wszystkie nagłówki dodawane są do przesyłek automatycznie przez program przygotowywujący list i wszystkie programy uczestniczące w jego przesyłaniu. Niektóre programy pozwalają definiować własne nagłówki i zawartość standardowych.

Nie będziemy tu dokładnie opisywać postaci i zawartości nagłówków MIME. Osoby tym zainteresowane powinny sięgnąć do odpowiednich dokumentów RFC. Jako podstawowe źródło informacji można polecić FAQ[7] grupy newsowej comp.mail.mime (dostępny również lokalnie[9]). Niezbyt (być może aktualna) wersja dostępna jest również jako „załącznik" tego dokumentu.

Wszyscy pragnący sprawdzić poprawność konfiguracji i zdolność do odbierania przesyłek MIME różnych typów mogą poprosić o to odpowiedni serwer[10].

O podobne dokumenty można również poprosić serwer pocztowy[11]. W polu „Subject" listu należy podać rodzaj próbki, który chcemy otrzymać.[12]

Proponowany przez standard MIME sposób kodowania pozwala na bezpieczne przesyłanie listów zawierających informacje „binarne" kanałami „zwykłej" 7–bitowej poczty. Opracowany został standard ESMTP pozwalający „negocjować" sposób przesyłania przesyłek „binarnych".

Pojawiają się pierwsze programy MDA[13] korzystające z tych możliwości. Niektóre z nich w przypadku komunikacji z agentem nie rozumiejącym ESMTP – dokonują **samodzielnie** jej „MIME–izacji" (dodając odpowiednie nagłówki i odpowiednio kodując przesyłkę).

W dalszej części przedstawimy dwa przykłady listów multimedialnych.

9.3.1 Przykład wiadomości zawierającej znaki „narodowe"

```
From:          "Wojciech Myszka" <myszka>
To:            myszka
Date:          Sun, 28 May 1995 09:21:27 MET-1METDST
MIME-Version:  1.0
Content-type: text/plain; charset=ISO-8859-2
Content-transfer-encoding: Quoted-printable
Subject:       test
X-mailer: Pegasus Mail v3.22

ma=B3e          wielkie
=B1  (a z ogonkiem)  =A1
=E6  (c z kresk=B1)    =C6
=EA  (e z ogonkiem)  =CA
```

[7] W chwili obecnej dwa polskie archiwa: „sunsite" w ICM w Warszawie[8] oraz w archiwach Cyfronetu w Krakowie zawierają komplety „FAQ–ów" wszystkich grup newsowych.

[9] ftp://sunsite.icm.edu.pl/pub/usenet/comp.mail.mime

[10] http://www-dsed.llnl.gov/documents/tests/email/emailform.html

[11] mailto:mime-test@relay.surfnet.nl

[12] Czyli ciąg znaków z zestawu: application/octet-stream, multipart/mixed, nested, application/wordperfect5.1, message/external, multipart/alternative, image/gif, multipart/digest, multipart/parallel, application/postscript, message/rfc822, image/jpeg, audio/basic, X-local, message/partial, application/msword, application/rtf, text/enriched, iso-8859-1, video/mpeg

[13] Mail Delivery Agent – program dostarczający pocztę; na komputerach UNIXowych będzie to najczęściej program **sendmail**.

```
=B3  (l z kresk=B1)    =A3
=F1  (n z kresk=B1)    =D1
=F3  (o z kresk=B1)    =D3
=B6  (s z kresk=B1)    =A6
=BF  (z z kropk=B1)    =AF
=BC  (z z kresk=B1)    =AC
=3D  (r=F3wno=B6=E6)
dluga linia dluga linia dluga linia dluga linia dluga linia dluga linia dl=
uga linia
```

Tak wygląda (przed rozkodowaniem) przesyłka zawierająca list zawierający „polskie litery”. Zostały one zakodowane z zastosowaniem algorytmu nazwanego „quoted-printable”. Polega on na tym, że znaki o kodach ASCII większych niż 127 zastępowane są znakiem równości „=” i kodem szesnastkowym znaku. Cechą charakterystyczną przesyłek mime jest też to, że wszystkie linie dłuższe niż 76 znaków są automatycznie łamane. Znak złamania linii zaznaczony jest znakiem „=”. Program odczytu poczty powinien dokonać rozkodowania.

Możliwe jest (w standardzie MIME) wysyłanie przesyłek zawierających „niezakodowane” znaki narodowe. Wówczas muszą pojawić sie nagłówki:

```
Content-type: text/plain; charset=ISO-8859-2
Content-transfer-encoding: 8bit
```

Ich brak może uniemożliwić (i bardzo często uniemożliwia) prawidłową interpretację znaków narodowych u odbiorcy. Stwarza też pewne kłopoty programom automatycznie MIME–izującym przesyłki.

9.3.2 Przykład wiadomości zawierającej informację tekstową i obrazek (plik .gif)

```
From myszka@ldhpux.immt.pwr.wroc.pl Thu Jun 12 07:46:00 1995
Received: from ldhpux.immt.pwr.wroc.pl by ldhp715.immt.pwr.wroc.pl with ESMTP
        (1.37.109.16/16.2) id AA158775998; Mon, 12 Jun 1995 07:46:39 +0200
Return-Path: <myszka@ldhpux.immt.pwr.wroc.pl>
Received: by ldhpux.immt.pwr.wroc.pl
        (1.37.109.16/16.2) id AA075205998; Mon, 12 Jun 1995 07:46:38 +0200
Date: Mon, 12 Jun 1995 07:46:38 +0200 (METDST)
From: Wojciech Myszka <myszka@ldhpux.immt.pwr.wroc.pl>
To: myszka@ldhp715.immt.pwr.wroc.pl
Subject: Test obrazka
Message-Id: <Pine.HPP.3.91.950612074541.7462A-100001@ldhpux.immt.pwr.wroc.pl>
Mime-Version: 1.0
Content-Type: MULTIPART/MIXED; BOUNDARY="2010929025-851401618-802935998=:7462"

  This message is in MIME format.  The first part should be readable text,
  while the remaining parts are likely unreadable without MIME-aware tools.
  Send mail to mime@docserver.cac.washington.edu for more info.

--2010929025-851401618-802935998=:7462
Content-Type: TEXT/PLAIN; charset=US-ASCII

To jest prosty test

--
Wojciech A. Myszka                          myszka@ldhpux.immt.pwr.wroc.pl
Technical University of Wroclaw, Institute of Material Science
and Technical Mechanics, Smoluchowskiego str. 25
50-370 Wroclaw, Poland, tel. (+48 71) 21-50-28, fax () 21-12-35
--2010929025-851401618-802935998=:7462
Content-Type: IMAGE/GIF; name="mouse.gif"
Content-Transfer-Encoding: BASE64
Content-ID: <Pine.HPP.3.91.950612074638.7462B@ldhpux.immt.pwr.wroc.pl>
Content-Description: Prosty obrazek
```

```
R0lGODdhQAHIAPcAAAgAABAICBgICCggIDgwMDgoKElBQUE4OFFJSVlJSWFZ
WWlhYXlxcXFpaYJ5eUFJSWlxcdvr80lhcSAoMFFZYWFpcWlxeXF5gqqyurK6
w7rDy8vT28PL09vj69Pb4+Pr80FJWThBUWFpeVlhcYKKmnmCkpqisrrD08PL
[...]
vkzI21vIrsLGcLyjpNT7NkR0RPECMapaSKnMqŚsy3J0xCIUszkkN8z8y7YQ
Ha60yUoERrDkSZckzK1ExsZUqJlTzPBRx9MSyIK7QMh8tu/awSS0N6y8vo1U
w9Qomjg8L7ZTS6HrsJWUTPHDS0xU05a8D+OkTNdcqKhyKgMkycf0TZJMzWDO
xWJKs4NwRhVTxB4kMlX7j890zpr7ShG7zIpsyvGCO/ZBTagiz4QzsoAAADs=
--2010929025-851401618-802935998=:7462--
```


Kolejny przykład przesyłki MIME jest nieco bardziej rozbudowany. Linia `Content-Type: MULTIPART/MIXED; BOUNDARY="2010929025-851401618-802935998=:7462"` mówi że w skład przesyłki wchodzi kilka części. Każda część jest **kompletną** przesyłką wyposażoną w swoje nagłowki określające dalej jej zawartość i sposób kodowania. Tekst „`--2010929025-851401618-802935998=:7462`" oddziela poszczególne części.

Poprawnie skonfigurowany program poczty multimedialnej pozwoli zapisać „binarną" część przesyłki do pliku, program dobry – pozwoli ją obejrzeć (uruchamiając odpowiednie narzędzie jeżeli są ku temu warunki), jeszcze lepszy program przedstawi ją w postaci przedstawionej na poniższym rysunku:

Γ Δ Θ Λ Ξ

Π Γ Δ Θ Λ

Ξ Π Σ Γ Δ

Pliki binarne zakodowane są z użyciem algorytmu „BASE64". Zwracam również uwagę, że kazdy tekst, który będzie zawierał więcej niż 11% (dokładniej 1/9) znaków binarnych (narodowych) może zostać potraktowany jako binarny i zakodowany z użyciem algorytmu Base64 co uczyni go praktycznie nieczytelnym dla odbiorców nie posiadających odpowiedniego oprogramowania. Algorytm kodowania opisany jest w dokumencie RFC1521.

9.3.3 Znaki narodowe w nagłówkach listów

O ile problem znaków narodowych w treści listu jest już całkiem nieźle rozwiązany w wielu „darmowych" produktach, o tyle obecność tych znaków w nagłówkach dla większości z nich stanowi poważny problem.

Zniekształcena w treści przesyłki spowodowane niezrozumieniem/złym zrozumieniem standardu MIME są stosunkowo niegroźne. Zniekształcenia w nagłówkach (zwłaszcza tych określających odbiorców poczty: To:, Cc:, Bcc:, ale też i nadawcę) mogą produkować niedoręczalne przesyłki.

Zasady pisania znaków narodowych w nagłówkach określa RFC1522 i przed wpisaniem własnego nazwiska z wszelkimi znakami diakrytycznymi zalecam przestudiowanie tego dokumentu i sprawdzenie, czy używany program pocztowy dobrze takie informacje przetransformuje. W cytowanym dokumencie znajdują się przykłady.

Najnowsze wersje programów **Pegasus Mail** (**pmail**) tak w wersji DOSowej jak i pracującej w środowisku Windows oraz **pine** pozwalają na poprawne kodowanie znaków narodowych w nagłówkach listów.

9.3.4 Konfiguracja poczty multimedialnej

Gdy mamy do dyspozycji sprawny i poprawnie skonfigurowany program odczytu poczty multimedialnej listy takie nie powinny sprawiać kłopotu. Co zrobić jednak, gdy tak nie jest?

Standardowy mechanizm rozkodowywania przesyłek MIME używany na komputerach UNIXowych wykorzystuje plik o nazwie `mailcap`. Korzystają z niego niektóre programy pocztowe i przeglądarki WWW. W środowisku Windows uznano najwyraźniej, że poprawna konfiguracja tego pliku jest zbyt złożona dla „zwykłego" użytkownika i nie jest on wykorzystywany.

Z przesyłkami MIME (jeżeli nie możemy ich odcyfrować inaczej) możemy próbować postępować w sposób następujący: zapisujemy przesyłkę do pliku **wraz z nagłówkami** (w programie **mail** lub **mailx** służy do tego komenda „copy"), a następnie przetworzyć ten plik za pomocą programu **metamail** (lub co sensowniejsze w środowisku UNIX przekazać bezpośrednio, za pomocą funkcji *pipe* do programu **metamail**).

Bardzo łatwo go zdobyć: źródła w wersji dla UNIXa znaleźć można w thumper.bellcore.com; jest tam też wersja skompilowana pracująca pod DOSem. Zawsze warto jednak skorzystać z Archiego[14] (szukając „**mm2.7**" i „**contrib2.7**").

Problem opisuje RFC1524 poświęcony właśnie konfiguracji programu odczytu poczty.

Zasadą (typowych) programów odczytu poczty multimedialnej jest to, że korzystają one z pliku konfiguracyjnego zwanego `mailcap`. Typowym miejscem, gdzie można go szukać to (jeżeli którykolwiek z dalszych odsyłaczy – ale tylko na komputerze UNIXowym – „zadziała" oznacza to, że jest, być może, na nim zainstalowany jakiś program ułatwiający czytanie poczty multimedialnej): `$HOME/.mailcap`, `/etc/mailcap`, `/usr/etc/mailcap`, `/usr/local/etc/mailcap`. Plik ten może być jednak umieszczony w innym miejscu – zakłada się wówczas, że zmienna środowiska `MAILCAPS` określa wszystkie te miejsca gdzie można tego pliku szukać.

Ale (niestety) nie wszystkie programy pocztowe korzystają z pliku `mailcap`. Pewne uwagi na temat konfigurowania programu **pmail** w środowisku MS–DOS zawarte są na stronie 10.2.1.

Natomiast z pliku tego korzystają bardzo intensywnie wszystkie *browsery* języka **HTML** wśrodowisku UNIX.

Jak już wspominałem na str. 9.3 nagłowkiem, który informuje o zawartości przesyłki jest nagłówek rozpoczynający się od ciągu znaków `Content-type:` . W pliku `mailcap` powinna znajdować się informacja jakiego programu należy użyć do „odtworzenia" danego typu przesyłki. Linia pliku `mailcap` może wyglądać tak:

```
text/richtext; richtext \%s; copiousoutput
```

Co oznacza, że przesyłki zawierające wiadomość tekstową (richtext wskazuje, że jest to wiadomość sformatowana z wykorzystaniem specjalnych znaków formatujących) należy odtwarzać z wykorzystaniem programu **richtext** (%s zostanie zastąpione nazwą pliku w którym program pocztowy (lub **metamail**) zapisze odkodowaną treść wiadomości).

Dokładniej, każda linia pliku `mailcap` podzielona jest na szerg pól oddzielaonych średnikami, dwa pierwsze pola są obowiązkowe, pozostałe nie i mogą występować w dowolnej kolejności. Oto **podstawowe** informacje na temat pliku `mailcap`:

Pierwsze pole opisuje zawartość przesyłki (to co pojawia się w nagłówku `Content-type:` listu), na ogół ma postać typ/podtyp, zamiast „podtyp" można umieścić gwiazdkę („*").

Drugie pole zawiera komendę, którą należy wykonać, aby „obejrzeć" wiadomość (lub jej fragment); dokładna postać tej komendy będzie zależała od typu systemu operacyjnego; miejsce, w którym ma być umieszczona nazwa pliku zawierającego wiadomość wskazujemy za pomocą znaków `%s`.

Kolejne pola mają nieco inną budowę: składają się ze słowa kluczowego, a po nim (opcjonalnie) wartości (po znaku równości). Słowem kluczowym może być (między innymi):

compose określa nazwę programu używanego do komponowania wiadomości określonego typu,

composetyped pole o bardzo podobnym znaczeniu jak „compose" (szczegóły w rfc 1524)

edit określa program używany do modyfikacji zawrtości określonego typu,

print program używany do drukowania zawrtości listu,

test pozwala wskazać program używany do sprawdzenia, czy zachodzą określone warunki, jeżeli nie – pozycja pliku `mailcap` nie jest brana pod uwagę, ale plik jest interpretowany dalej,

needsterminal pole informuje program (pracujący w środowisku okienkowym) że wyświetlenie listu wymaga uruchomienia interakcyjnego terminala,

[14] http://www.pvv.unit.no/ftpsearch/

copiousoutput mówi, że wyjście z programu używanego do czytania przesyłki będzie ciągiem bajtów (znaków) i jest rzecczą programu pocztowego „przepuszczenie” ciągu znaków przez jakiś program stronicujący,

description pozwala umieścic opis.

Załóżmy, że otrzymaliśmy list z nagłówkiem `Content-type: text/x-html` zawierający jako treść plik zawierający komendy języka HTML.[15] Pragniemy użyć programu **lynx** do przeczytania (obejrzenia) tego tekstu. Zapis w pliku `mailcap` o postaci:

```
text/x-html; lynx -force_html %s
```

może pozwolić „odebrać” taki list.

Szereg informacji na temat takiego konfigurowania różnych systemów oprogramowania (w tym również poczty) aby można było sensownie korzystać ze znaków narodowaych znalezc można w Krakowie.[16]

Przykład: co zrobić, żeby zawsze poprawnie widzieć polskie litery

Załóżmy teraz, że chcemy tak skonfigurować system, aby w zależności od warunków w jakich pracujemy poprawnie czytać list z „polskimi” literami.

Co nam do tego będzie potrzebne? Wspomniany już (patrz str. 9.3.4) program **metamail**. Jak wspominaliśmu dostępny jest on i dla środowiska DOS.

Przygotować musimy też plik `MAILCAP`. Zakładam, że plik już istnieje i będziemy go tylko modyfikować.

Załóżmy dalej, że nie mamy żadnej możliwości wyświetlenia polskich liter na terminalu.

Załóżmy również, że zgodnie ze standardem MIME przesyłka zawierająca polskie znaki ma nagłówek:

```
Content-type=text/plain ; charset=iso-8859-2
```

W pliku `MAILCAP` modyfikujemy (jeżeli już jest) lub dopisujemy (jeżeli nie ma) następującą linię:

```
text/plain ; nopl %s
```

Tworzymy również plik wsadowy (skrypt) o nazwie `nopl` i umieszczamy go w którejś z kartotek znajdujących się w ścieżce `PATH` zawierający polecenia zamieniające znaki o kodach ISO-8859-2 na zwykłe ASCII. Może to być coś takiego:

```
plconv -ISO-Latin-2 <$1 | plconv +brak |less
```

(Ja wykorzystałem program **plconv**[17] Jarka Strzałkowskiego.)

Analogiczne rozwiązanie można zastosować i w środowisku DOS (w tym przykładzie zakładam, że używany jest interpreter poleceń 4DOS). Plik `nopl.bat` wygląda tak:

```
if %_CODEPAGE EQ 852 (plconv -ISO-Latin-2 <%1 | plconv +IBM-Latin-2 |less)
if NOT %_CODEPAGE EQ 852 (plconv -ISO-Latin-2 <%1 | plconv +brak |less)
```

(Wykorzystałem dodatkowo fakt, że jeżeli została zainstalowan strona kodowa 852 – polskie litery można spokojnie wyświetlać.)

Cały problem polega na tym, że większość znanych mi programów pocztowych zakłada, że „umie” poprawnie wyświetlać przesyłki o zawartości „`text/*`” i w tych przypadkach niechętnie korzysta z z pliku `MAILCAP`. Oczywiście zapisanie treści listu (wraz z nagłówkami) do pliku i przetworzenie go za pomocą programu **matamail** powinno dać oczekiwany skutek.

Wykorzystnie parametru `test` pliku `mailcap` pozwala budować znacznie bardziej rozbudowane konstrukcje dokonujące różnego rodzaju przekształceń tekstu w zależności od tego jaki zestaw znaków jest używany.

Możemy również przekazać pewne dodatkowe parametry do uruchamianaego programu. Jeżeli w pliku `mailcap` pjawi się linia następująca:

```
text/plain; nopl %s %{charset}
```

podczas wywołania programu `nopl` ciąg znaków `%{charset}` zostanie zastąpiony przez aktualną wartość parametru `charset` czytanej przesyłki.

[15] Hypertext Markup Language
[16] http://www.uci.agh.edu.pl/ogonki/
[17] http://www.agh.edu.pl/ogonki/cos

9.3.5 Jak wysłać plik binarny gdy nie mamy dostępu do poczty multimedialnej?

Sprawa jest stosunkowo prosta. Trzeba zastosować jeden z wielu programów kodujących pliki binarne do postaci znakowej. Do najpopularniejszych należą:

uudecode/uuencode wywodzący się z systemów UNIXowych i dostępny również na innych platformach,

BinHex wywodzący się ze środowiska komputerów Apple.

(Jest również kilka innych algorytmów, ale ich mnogość raczej utrudnia zadanie, zwłaszcza odbiorcy tak zakodowanych przesyłek.)

Polecam program **uuencode** służący do kodowania do postaci tekstowej i **uudecode** rozkodowywujący do postaci binarnej.

Załóżmy, że chcemy przesłać plik `mouse.gif`. Sekwencja rozkazów w systemie UNIXowym jest następująca:

```
uuencode mouse.gif mouse.gif | mail user@address
```

a więc bardzo prosta. Znaczenie jest następujące:

1. uruchamiany jest program **uuencode** czyta plik `mouse.gif` (pierwszy parametr) i kodowany jest do postaci znakowej (drugi parametr `mouse.gif` mówi pod jaką nazwą plik zostanie zapisany po rozkodowaniu),

2. Zakodowana zawartość pliku wypisywana jest na standardowym wyjściu, a przechwytuje ją dzieki znakowi `|` program **mail**,

3. Program **mail** po przeczytaniu całego pliku ekspediuje go na wskazany adres.

Podstawowym ograniczeniem takiego przesyłania plików (mimo pozornej łatwości) jest to, że nie zwracamy uwagi na wielkość tak ekspediowanych plików. Wielkie przesyłki (w przypadku wolnych łącz) przesyłane są bardzo długo. Niektóre węzły pośrednie (lub nawet docelowe) mogą mieć wbudowane ograniczenia na wielkość przyjmowanej przesyłki.

Tak przesłany list u odbiorcy powinien być zapisany do pliku, a następnie rozkodowany za pomocą programu **uudecode**. Jest on tak skonstruowany, że pomija wszystkie pocztowe nagłówki i inne teksty występujące przed zakodowaną zawartością.

Listy takie najłatwiej „rozkodowywać" wykorzystując dostępną w wielu programach pocztowych opcję *Pipe*, na przykład `|` w programie **pine** przekazując zawartość komendzie **uudecode**.

Program **pmail** automatycznie rozpoznaje listy zawierające zakodowane w ten sposób załączniki i sam „proponuje" rozkodowanie ich i zapisanie do pliku.

Rozdział 10

Opisy popularnych programów pocztowych

Wielość programów pocztowych i różnorodność zainteresowań użytkowników uniemożliwia praktycznie opracowanie sensownego zestawu instrukcji. Dlatego też nie zamierzam tego robić.

Pozwalam sobie jedynie umieścić tu odsyłacze do opisów, które znalazłem „gdzieś” w sieci.

Niestety, różni autorzy przyjmują różne założenia co do sprzętu i możliwości czytania plików HTML przez użytkowników. Stąd, może zdarzyć się, że niektóre z dokumentów przystosowane są do czytania w środowisku graficznym.

Warto zaglądać również, do różnych opisów programów pocztowych, które pojawiają się w czasopismach ([19]) czy omówień oprogramowania dostępnych w sieci (tu polecam *mirror* – lusterko archiwum oprogramowania korzystającego z Winsock[1 2])

10.1 Program pine

Po pewnym okresie używania programu **elm** „natknąłem” sie na program **pine**. Prawdę mówiąc nie zachwycił mnie. Później, gdy dokładniej przyjrzałem się jego cechom:

1. możliwość wysyłania i odbioru przesyłaek MIME,
2. możliwość czytania i wysyłania newsów,
3. możliwość przeglądania z jednego miejsca poczty na kilku komputerach,
4. bardzo prosty, przejrzysty i konsekwentny interfejs użytkownika

doszedłem do wniosku: to jest właśnie to!.

Program zainstalowałem na swoim komputerze i polubiłem go. Widząc zmagania początkujących użytkowników poczty z programem **mail** polecilem im **pine**. W końcu postanowiłem napisać coś w rodzaju podręcznika. Bardzo długo stanowił on część tego opracowania. Gdy jego rozmiar rozrósł się – postanowiłem oddzielić go od narzędzi sieciowych. Dostępny jest jako osobny dokument.[3]

10.2 Programu pmail – Pegasus Mail

Γ

Niestety nie dorobiłem się własnego opisu tego programu, choć sam go używam (w domu, w połączeniu z pocztą UUCP) i uważam za najlepszy, znany mi program obsługi poczty pod DOSem.

[1] http://www.man.poznan.pl/hypertext/cwsa/mail.html

[2] Niestety, cały przegląd bardzo mocno korzysta z bardziej zaawansowanych elementów języka HTML – należy go czytać za pomocą jakiegoś Netscape'a :-(

[3] http://www.immt.pwr.wroc.pl/export_hp/pine/pine.html

Mimo, iż używany jest głównie w środowisku sieci Netware Novell, został tak pomyślany i zrobiony, że może być stosunkowo łatwo skonfigurowany do pracy z wykorzystaniem praktycznie dowolnego sposobu transmisji listów z serwera.

Idea funkcjonowania programu jest taka:

1. Poczta przychodząca zapisywana jest we wskazanej kartotece użytkownika; każdy list zapisywany jest jako osobny plik.

2. Program przegląda tę kartotekę i jeżeli są tam jakieś pliki traktuje je jako nowe listy. Listy przechowywane są przez program w skrzynkach pocztowych (folderach) w jego własnym formacie.

3. Listy wychodzące są albo zapisywane w kolejnej kartotece (kolejka poczty wychodzącej) i wysyłane „później", albo przekazywane natychmiast po napisaniu programowi, który je ekspediuje w świat.

Taka konstrukacja pozwala przystosować program do współpracy z praktycznie dowolnym systemem przesyłania poczty na serwer: POP, UUCP, SMTP, dyskietka :-)

Mogę polecić jego dokumentacje[4] dostępną na gopherze Centrum Informatycznego Politechniki Wrocławskiej, a opracowaną przez Ryszarda Bernardyna i Izydora Statkiewicza.

„Próbna" wersja HTML (przetworzona z powyższego tekstu) jest również dostępna.[5]

Pewne moje dodatkowe uwagi na temat konfigurowania tego programu do odczytu poczty multimedialnej i wykorzystania jako czytnik poczty „off–line" podaję niżej.

10.2.1 Konfiguracja programu pmail do odbioru poczty multimedialnej

Sposób dostosowania programu **pmail** do wysyłania poczty polskojęzycznej został opisany wraz z innymi informacjami (na podstawie różnych źródeł „sieciowych") przez Piotra Piątkowskiego.[6]

Program **pmail** nie korzysta z pliku `mailcap` (patrz str. 9.3.4). Jego odpowiednikiem jest plik `pm-mime.r` znajdujący się w standardowej dystrybucji programu. (Choć zapewne nie jest on instalowany przez wszystkich administratorów.)

Cechą charakterystyczną wiadomości MIME są nagłówki zawierające informacje o zawartości zbioru i o sposobie kodowania. Szczegóły podaje RFC1521. W pliku `pm-mime` padajemy programowi **pmail** jakich narzędzi ma użyć do „przeglądania" kodowanych wiadomości.

I tak na przykład do przeglądania obrazków (`Content-Type: image/gif`) używam dostępnego w pakieci **NC** programu bitmap, do przeglądania plików „zzipowanych" (`Content-Type: application/zip`) programu **shez**.

Oto odpowiedni fragment pliku `pm-mime.r` („8" na początku struktury informuje ile pozycji danych w strukturze jest „aktywnych"):

```
mime_handlers resource_0
   {
   8,  ## Note that this means only the first 8 entries are valid viewers

   0, "IMAGE",       "GIF",           "c:\\nc\\bitmap.exe ~c",
   0, "TEXT",        "ENRICHED",      "richtext ~c|less"
   0, "TEXT",        "RICHTEXT",      "richtext ~c|less"
   0, "APPLICATION", "WORDPERFECT5.1", "wpview ~c",
   0, "APPLICATION", "MSWORD",        "wpview ~c",
   0, "APPLICATION", "ZIP",           "shez ~c",
   0, "APPLICATION", "X-pgp-message","pw ~c"
   0, "APPLICATION", "pgp",           "pw ~c"
   0, "APPLICATION", "ODA",           " ",
   0, "IMAGE",       "JPEG",          " ",
   0, "VIDEO",       "MPEG",          " ",
   0, "APPLICATION", "POSTSCRIPT",    "call ala.bat ~c",
   0, "APPLICATION", "OCTET-STREAM", "bview ~c",
   1, "AUDIO",       "BASIC",         "i:\\scratch\\4\\auplay.bat ~c",

   #  You may add other MIME types to this list as they are defined.
   };
```

[4] gopher://sun1000.ci.pwr.wroc.pl:70/00/wcss/inforci/Informator19
[5] http://www.immt.pwr.wroc.pl/~myszka/pmail/pmail.html
[6] http://www.uci.agh.edu.pl/ogonki/

(Zero w pierwszej kolumnie tabeli w tym pliku informuje, że użytkownik zostanie zapytany czy:

- chce wiadomość obejrzeć za pomocą odpowiedniego narzędzia (po rozkodowaniu),
- chce zapisać do pliku,
- czy też rozkodować i mimo wszystko obejrzeć za pomocą **pmaila**);

jedynka mówi że nie będzie pytany – zostanie użyte zadeklarowane narzędzie do "obejrzenia" wiadomości.

10.2.2 pmail a PGP

Niestety, mimo usilnych starań nie mogę znaleźć sposobu łatwej integracji programu **pmail** z systemem kodowania informacji PGP.[7] Nie ma też żadnego standardu wysyłania przesyłek kodowanych za pomocą PGP. Niektóre programy (na przykład **elm** ver. 2.4) kodują takie listy jako przesyłki MIME. Zawartość określana jest jako `application/X-pgp-message`. Pozwala to użyć zewnętrznego programu do rozkodowywania (sprawdzania podpisów). Serwery używane do „rozpowszechniania" kluczy publicznych za pomocą poczty „pakują" klucze do przesyłek MIME `application/pgp`.

W poprzednim rozdzial (patrz str. 10.2.1) zamieściłem przykładową zawartość pliku `pm-mime.r` określającego sposób zachowania programu **pmail** w różnych sytuacjach, w tym również w przypadku napotkania poczty kodowanej (lub podpisanej) za pomocą PGP, a "pakowanej" w przesyłkę MIME.

Gdy jednak otrzymujemy list kodowany (lub podpisany za pomocą) PGP, który nie jest zakodowany jako przesyłka MIME, a jego zawartość wygląda na przykład tak:


```
-----BEGIN PGP MESSAGE-----
Version: 2.6.2

hIwDiIljtbUmTZUBA/wJRQfB4dXQkKMKZn2oDhL6sTg+Um/2Oh8c+EMGe/C54CJ/
93ZcnS7pHMEetSvBFRHC8nf2nXKd0Ww4Z9c6TNLSrAZhbpBuyEnbabjNQ+63sdmU
HR5UM1CdN8jyvfiXXmJxZa9odbPjPeAd8GzslwEejbTqEXFnZ2mxTJCuaai8qISM
Axsig5ZWoVINAQQAkdjpp8N8c+XVpfnBeaHq9Ho+B5mCOlM5ZErJQUo48zqsZCNl
OIms0tJ4d/AYA/shXoCyMxvjFFGGC+5NjvqaEWAW3Ca5+GUAAvZ4KC6Puj13SGVN
HoGD4SVID+uFYms8SrPd+aLQKRBJie7Xa3JShBbvS0QdtKx4IA2NcGNO+RqEjAPO
SYrNs9ri8QED/2SGsZ1pYK2oVRaDI7enzt1V5HjBgqXpm9Ydf6oVgF7D+/nfj5W
aK/6KOefs3WsY1ANinJbEe473kcIyCz9N/gctCMx3wSyKMQqbSTEGZIvEIABN5ZK
QAf/E5hcoHZ+Js9Mqb4uuqBnvDdIvaxzApiivubfR+RAXQ/SrNicNu3ShGwDl5hb
CbULY/OBAvOUKuEfWObYZ/AVy3Bkr0a0ebo7ocMvHpCe9n4RfnEUVxCQG5S++OHW
CgFEov3swjaCP1H5mkX7NC3QloKi5GDjWOgC9FVC1zu0SirbjeKcIf5yHvqrRnkx
HY7VQnR/4humAAAC5Ygw/athNkbTLANDiGz9jwyRVreHDwW0rGJ6uQKZjiqmmoMN
499z+zEsv2UAuy4sW3mbaPaXXfMEPGKQsnjfdA8ub9evzHpl6VNrwm/kcofyMAFA
Q7DJaajBhhSUYxkE0hBXyJJsvaTHq0tYt/nkixHBWOp23smeO6nx8loNUmpU2Eie
y1cKJi73728TCqEwt5svrtenTDtM3KoxCioFSNkxNlyOBa1RDkKfvk2ZwB3Jucyx
825OUz9mw5kklzee5q0ScJrpb8WZaQFzIcHRxHyrh5YL4lnP/KpwlA42oIFMtH7W
ARwz8u4CsIV3qbtG3LIO3qzGm83fs7IOCiqIXuR3y7nO9/i395gMeypHeFeueHlz
5eEy9mdcDhjWPnbmLfilFDn0ewtLfG4mj3mSt0tzy+ttbja5aqeWb/eIc3udgtZc
jhteiazH2BEiour0zSluJ7Jhod5lZ3tNCQqrviEeILkqISiZQS43QOruQT65G8en
F2Qn7xZFi1j4dAY7l8gIBSxYQBypoagLLJl//UhUEWlI7xYTCMFOHp41KDkuHeqc
wHBVohx5o1SsRvmf9cx3oVvdwunyOZofg3hW3j9t4X07twx+4JDUILpMdUbeVqk6
jRbysQD8VGfmlslmEgersbyOo1urQz+1ZpieM/s6LjNu5gze9Czmxb7Ag+xs1dg7
vpoAL8LF3cxqkvEKxDjK+sIvI+RLLrSu2ptmt5/yA516vlYL/H6fYBuKWslrJ7MT
SkmbD1f8hdqCHtySYNdYHBFkvBrDMP3BvKd7Nc5aeCSiAIrOZScMWiSxCNsp0fkp
fqVMwfL8q/Uf5llwxuLkQWVXI6ppmEpBWYYyrEXZTM0sd6D/AVicJ5ePFcjyl5es
zBlSiGnPLRGG22gvCMbstuBYXTuimZRnW++0Z2A+y9iCvTh0a6Npa1N6BdPGEkCO
JvD270jrkWFc9hB1EWeA5X9OoB7SBM6hCGSp+NO0JtdSrA==
=q25Q
-----END PGP MESSAGE-----
```


sytuacja jest nieco trudniejsza, ale nie tragiczna.[8] Wystarczy wskazać wiadomość kursorem, nacisnąć klawisz **D** i w okienku wpisać „**pgp -m**". Zawartość wiadomości zostanie przekazana do programu **pgp** (zakładam,

[7] Więcej informacji na temat PGP można znaleźć w opisie Szymona Sokoła.

[8] Wiadomość została zakodowana dla czterech osób, i tylko one mogą ją odczytać.

że użytkownik ma ten program i został on poprawnie zainstalowany) który (jeżeli tylko list jest adresowany i zakodowany do nas) rozkoduje go prosząc o podanie hasła (lub sprawdzi podpis).

Jezeli chodzi o wysyłanie – problem jest nieco trudniejszy. Na razie trzeba list przygotowywać dowolnym edytorem, a następnie kodować (lub podpisywać) używając **pgp**. Tak przygotowany plik wczytujemy do edytora podczas przygotowywania listu. **Zwracam uwagę, że zakodowanych (lub podpisanych) przesyłek nie można już poprawiać lub modyfikować w jakikolwiek sposób!** Może sprawić to peweien problem, gdzyż edytor **pmail**a automatycznie łamie wiersze dłuższe niż zadana szerokość.[9]

Można również wykorzystać program **PGPWave**[10] autorstwa Johna Stephensona. Został on zaprojektowany jako dosyć prosty interfejs do programu **pgp** stosowany w programach czytania poczty *offline*. Należy on do kategorii oprogramowania *FREEware*.

Program trzeba (po skopiowaniu do jakiejś kartoteki) skonfigurować. Musimy zdecydować się w jakiej kartotece założymy plik konfiguracyjny pamiętający wszystkie ustawienia. W pliku `pw.bat`, w linii powidującej uruchomienie programu **PGPWave** jako drygi parametr trzeba podać dokładną lokalizację pliku konfiguracyjnego:

```
C:\PGP\PGPWAVE\PGPWave %1 C:\PGP\PGPWAVE\PGPWAVE.CFG
```

Program sam utworzy „wzorcowy” plik konfiguracyjny, jednak o poprawność ustawień musimy zadbać sami. W tym celu z menu wybieramy pozycję **Setup parameters** i wpisujemy ścieżki dostępu do wszystkich programów użytkowych z których chcemy korzystać.

Jeżeli zdecydujemy się na korzystanie z tego programu wskazujemy go jako „zastępczy” edytor dla programu **pmail.**[11] Po uruchomieniu program zgłasza się ładnym menu, które pozwala przygotować tekst listu (korzystając z zewnętrznego edytora), podpisać, zaszyfrować, podpisać i zaszyfrować list (nawet do 20 odbiorców). Wadą programu jest to, że nie udostępnia żadnych komend zarządzania kluczami.

Program może być wykorzystany również do czytania listów. Postępujemy tak jak opisano poprzednio: zamiast podawać jako zewnętrzne polecenie **pgp -m** wpisujemy **pw**.

Można próbować wykorzystać jeden z wielu innych dostępnych programów używanych jako przyjazny „front–end” dla systemu PGP. Wiele z nich pozwala uprościć wiele czynności (kodowanie, dekodowanie, podpisywanie, operacje na kluczach na „kółku publicznym” i prywatnym).

10.2.3 Jak wykorzystać program pmail do czytania/wysyłania poczty off–line?

Załóżmy, jesteśmy w następującej sytuacji:

- mamy (w domu, na przykład) modem i program komunikacyjny (na przykład **kermit**),
- mamy konto na komputerze UNIXowym, do którego mamy dostęp z wykorzystaniem modemu,
- chcemy zaoszczędzić parę groszy (a nie jesteśmy ogromnie sprawnym użytkownikiem UNIXa i ten fakt bardzo nas deprymuje powodując wiele pomyłek i wydłużanie czasu połączenia telefonicznego).

Czasami problem może być nieco inny: nazbierało się nam dużo poczty, której nie mamy czasu czytać w pracy; chcielibyśmy nagrać wiadomości na dyskietke, przenieść do domu i tam spokojnie odczytać, przygotować odpowiedzi i wysłać je później (korzystając z modemu lub niosąc na dyskietce do pracy).

Do rozwiązanie tego problemu wykorzystuje się programy odczytu poczty *off-line* (to znaczy bez połączenia z komputerem pracującym w sieci). Jest ich co najmniej kilka (czy nawet kilkanaście). W tym miejscu pokażemy jak wykorzystać znakomity program pocztowy pracujący w środowisku DOS/Windows – Pegasus Mail (**pmail**).

Jakie czynności trzeba wykonać:

1. Musimy zainstalować na domowym komputerze odpowiednią wersję programu: pracujący pod DOSem lub pod MS-Windows.
2. Musimy go odpowiednio skonfigurować w trybie samodzielnym *standalone*.

Po krótce opiszemy obie te czynności.

[9] Teksty **kodowane** z użyciem **pgp** są przekodowywane do postaci ASCII i długość wiersza otrzymanego tekstu nie przekracza 64 znaków, w przypadku tylko podpisywania – tekst zachowuje oryginalną długość wierszy.

[10] Dostępny jest on w archiwach SimTel (ja korzystam z najblizszego)

[11] W menu głównym programu **pmail** wybieramy „**P**references”, a następnie „**E**ditor and keyboard settings”. Jako „**E**ditor string” podajemy `pw %s` (można wskazać pełną ścieżkę). W zależności od upodobań możemy ten „edytor” wykorzystywać zawsze lub tylko na żądanie (ja wybieram to ostatnie rozwiązanie).

Instalacja programu

Jak zwykle, w takich sytuacjach musimy zdobyć **pmail** lub **winpmail** (w zależności od naszych preferencji). Jeżeli w naszej instytucji używana jest sieć Novell jest duże prawdopodobieństwo, że lokalny administrator ma gdzieś programy, jeżeli nie – musimy poszukać używając archiego i szukając ciągu znaków „pegasus" (przy okazji możemy poszukać dodatkowego narzędzia dokonującego konwersji folderów UNIXowych do standardu **pmail** szukamy ciągu znaków „unx2pm").

Gdy już mamy potrzebne pliki (są to pliki skompresowane) tworzymy kartotekę w której je „rozpakowywujemy" (wersja windowsowa, której używałem była spakowana „podwójnie" – te drugie archiwum musimy rozpakować tak aby zachować strukturę kartotek: `pkunzip -d`).

Przed rozpoczęciem konfiguracji możemy zapoznać się z dokumentacją programu. Podstawowa jej wersja dostępna jest w postaci programu **guide**. Można też poczytać cytowaną już wcześniej dokumentację.

Następnie uruchamiamy program **pconfig** i z menu wybieramy odpowiednią wersję instalacji: dla DOSa lub dla Windows. Z kolejnego menu wybieramy opcję „Standalone Configuration". W pierwszym podejściu zgódźmy się na wszystko co proponuje program instalacyjny, jak poznamy **pmail** lepiej – będziemy mogili go „przeinstalować".

W przypadku wersji MS–Windows należy dodatkowo przekopiować wszystkie biblioteki (pliki `*.dll` do kartoteki znajdującej się w „ścieżce" *PATH*).

Konfiguracja

Teraz część najważniejsza: konfiguracja programu. Musimy jedynie skonfigurować „bramkę" (*Manage* **u***ser defined gateways*). Wybieramy tę opcję, powinno się otworzyć okienko zatytułowane *Defined gateways:*. Naciskamy klawisz Ins żeby dodać nową bramkę. Pojawia się formularz, który powinniśmy wypełnić:

```
           Gateway name : [                 ]
         *New mail path :
   Is ^ a program to run? :  N
    *New mail search mask :
      *Outgoing mail path :
   *Run for outgoing mail :
        *Filename format :  ~d~d.CNM
  Run to validate address :
    *Reply address format :
  Accepts SMTP addresses? :  N
  Simple message headers? :  No formatting
    UUEncode attachments? :  N
          Burst messages? :  N    Gateway processes BCC? : N
      Strip gateway name? :  N
  Force all mail through? :  N
```

Omówimy go poniżej:

Gateway name podajemy tu nazwę bramki, możemy mieć zdefiniownych ich kilka, aby wymusić wysłanie poczty przez określoną bramkę adres podajemy w postaci „gateway/address";

***New mail path** wpisujemy tu nazwę kartoteki, w której program powinien się spodziewać „nowej" poczty;

Is ^ a program to run? w odpowiedzi wpisujemy **N** (w przypadku bardziej złożonych bramek można używać programu sprowadzającego pocztę z serwera);

***New mail search mask** określamy jak będą wyglądały pliki zawierające „nowe" listy – proponuję wpisać `*.cnm`;

***Outgoing mail path** tu podajemy kartotekę, do której będą wpisywane „wychodzące" listy;

***Run for outgoing mail** pole to pozostawiemy puste, w bardziej złożonych sytuacjach podaje się tu nazwę programy, który dostarcza pocztę na serwer;

***Filename format** proponuję zostawić proponowaną przez program wartość: `~d~d.CNM`;

Run to validate address pozostawiamy pusty (można tu wpisać program, który będzie sprawdzał poprawność adresów wychodzących listów);

***Reply address format** należy tu wpisać adres który będzie identyczny z adresem naszego serwera, to znaczy: jeżeli poczta wysyłana z serwera ma w nagłówku `From:` „myszka@ldhpux.immt.pwr.wroc.pl" taką wartość wpisujemy w tym polu;

Accepts SMTP addresses? wpisujemy **Y**;

Simple message headers? pozostawiamy tak jak proponuje program: `No formatting`;

UUEncode attachments? proponuję wpisać **N**;

Burst messages? proponuję zostawić **N** – w przypadku gdy zmienimy na **Y** **pmail** będzie tworzył osobny plik z treścią przesyłki dla każdego odbiorcy;

Gateway processes BCC? ptanie pojawiło się w nowszych wersjach programu; znaczy ono tyle „Czy oprogramowanie bramki bierze na siebie obsługę adresów **BCc:**"[12] trzeba odpowiedzieć **Y**. **sendmail** bowiem jest na natomiast tyle inteligentny, że gdy zobaczy pole **BCc:** – list na ten adres wyśle, ale tego nagłowka w wysyłanych listach nie umieści.

Strip gateway name? proponuję wpisać **Y**;

Force all mail through? proponuję pozostawić **N** wówczas będziemy mogli korzystać z poczty lokalnie (na komputerze PC), gdy odpowiemy **Y** wszystkie listy będą przesyłane z wykorzystaniem tej bramki.

O tym jak skonfigurować program **pmail** aby można było pisać i czytać po polsku już kiedyś napisałem. Nie wiem czemu (ale zapewne z lenistwa) muszę zrobić odsyłacz do tego tekstu znajdujący się gdzieś w Krakowie.[13]

Czytanie poczty off–line

Aby czytać pocztę należy przenieść z serwera zawartość skrzynki pocztowej. Znajduje się ona zwykle w kartotece `/usr/mail` (lub `/usr/spool/mail`) i ma nazwę zgodną z nazwą użytkownika. Plik ten trzeba skopiować na komputer „domowy" i przekształcić do postaci rozpoznawanej przez program **pmail**. Do przekształcania służy program **unix2pm**: `unix2pm -s mbox -o xxxx` gdzie:

mbox nazwa pliku zawierającego kompletny folder (powinien być on skopiowany w trybie binarnym),

xxxx nazwa kartoteki wskazanej programowi **pmail** jako miejsce zawierające „przychodzącą" pocztę.

W ten sposób można czytać wszystkie foldery utworzone przez programy **elm**, **pine** (komendą save), czy nawet **tin**[14] (o ile artykuły zostaną zapisane do pliku o nazwie „`=`").

Podczas czytania można używać wszystkich opcji programu **pmail** (łącznie z komendami **reply**, **compose** czy **forward**). Wszystkie wychodzące listy będą zapisywane jako pliki o rozszerzeniu `.cnm` w kartotece przeznaczonej na listy „wychodzące".

Wysyłanie poczty

Aby wyekspediować wszystkie listy należy wszystkie pliki o rozszerzeniu `.cnm` znajdujące się w kartotece poczty „wychodzącej" skopiować (czy raczej przenieść – po skopiowaniu kasujemy je oczywiście z kartoteki) (dowolną metodą) na serwer i wyekspediować w świat za pomocą następującej komendy (wykonywanej na serwerze):

```
for a in *.cnm
do
 /usr/lib/sendmail -t <$a
 rm $a
done
```

(Oczywiście musimy zorientować się, gdzie na naszym komputerze znajduje się program **sendmail**.) Komenda **rm** została użyta aby nie ekspediować wielokrotnie tych samych listów. Możemy oczywiście utworzyć odpowiedni plik zawierający te komendy i nadać mu atrybut wykonywalności.

[12] Blind Carbon copy – kopia listu, o której nie będzie wiedział żaden z odbiorców poczty.
[13] http://www.uci.agh.edu.pl/ogonki/pmail.html
[14] **tin** to popularny program do czytania newsów na komputerach UNIXowych.

10.2.4 Pmail i POP

Program **pmail** może współpracować z serwerem poczty POP. Nie będziemy tu opisywali zasad funkcjonowania serwera poczty POP. Jest to w chwili obecnej jeden z najbardziej popularnych sposobów przesyłania poczty z serwera na komputer osobisty użytkownika.

Idea jest taka: wszystkie listy skierowane do nas przychodzą na serwer pocztowy pracujący „na okrągło" i są tam magazynowane do czasu, aż użytkownik nie skontaktuje się z nim i nie prześle poczty na swój komputer. Najczęściej listy przesyłana na komputer osobisty są z serwera kasowane (choć można zadysponować inaczej).

Aby w praktyce zrealizowac tę ideę potrzebny jest program, który listy sprowadzi z serwera, a następnie przygotowaną przez nas pocztę wyekspediuje.

Autor programu **pmail** David Harris opracował odpowiednie narzędzie: nazywa się ono **pmpop** i powinien być dostępny w każdym pełnym archiwum programu **pmail**. O ile wiem najbliższym jest serwer w Wideniu[15]. Program dostępny jest na zasadach identycznych jak **pmail**.

Instalacja programu jest stosunkowo prosta:

1. Po skopiowaniu na komputer pliku archiwum, należy je „rozpoakować" i wybrać wersję programu, z której będziemy korzystać; dostępne są dwie:

   - współpracująca z jądrem TCP/IP LAN WorkPlace for DOS (nazwana lwpmpop.exe),
   - wymagająca do pracy jedynie programu *packet driver* odpowiedniego do używanej karty sieciowej (pdpmpop.exe).

   Wybrany program kopiujemy (pod nazwą pmpop.exe) do kartoteki znajdującej się w ścieżce `PATH`.

2. Należy skonfigurować parametry „sieciowe" (nie dotyczy przypadku, gdy korzystamy z wersji pracującej z LAN WorkPlace). Polega to na wypełnieniu pliku `WATTCP.CFG` informacjami właściwymi dla sieci, w której pracuje komputer. Potrzebne są następujące parametry:

   my_ip numer IP komputera,

   netmask maska sieciowa używana w sieci lokalnej (cztery liczby o postaci xxx.xxx.xxx.xxx),

   nameserver numer IP komputera spełniającego funkcję serwera DNS (bezpiecznie jest podać dwa takie serwery w dwu kolejnych liniach),

   gateway numer IP routera łączącego sieć ze światem zewnętrznym,

   domainslist lista domen, które będą dodawane do każdej nazwy komputera w procesie przekształcania nazw sympolicznych na numery IP. (Patrz również uwagi w rozdziale poświęconym adresom sieciowym (str. 5.2)).

   Plik umieszczamy albo w kartotece, do której skopiujemy program **pmpop**, albo w dowolnej kartotece – wówczas nadajemy zmiennej środowiska `WATTCP.CFG` wartość równa nazwie tej kartoteki.

3. Kopiujemy pliki `PMGATE.SYS` oraz `PM-MENU.RSC` do kartoteki, w której znajduje się program **pmail**.

   Uwagi:

   - Jeżeli już korzystamy z funkcji *User Defined Gateway* programu **pmail** powinniśmy zdefiniować nową bramkę zgodnie ze wskazówkami zawartymi w dokumentacji programu. (Procedura jest zbliżona do tej opisanej podczas definiowania bramki wykorzystywanej przez pocztę UUCP.)
   - Podobnie, jeżeli już utworzyliśmy własne menu, należy zmodyfikować odpowiednie pliki zgodnie ze wskazówkami zawartymi w dokumentacji.

4. Uruchamiamy **pmail** i z menu głównego wybieramy funkcję **Check for new mail**, i dalej **Configure PMPOP**. W otwartym oknie podajemy następujące informacje:

   Get Mail From wstawiamy nazwę serwera pocztowego,

   User name nasz identyfikator na tym komputerze,

   User password hasło na serwerze,

   SMTP Mail host nazwę serwera odbierającego wysyłaną pocztę (w absolutnej większości przypadków będzie to ten sam komputer co serwer pocztowy, na który przychodzą listy).

   Naciskamy klawisz Ctrl–Enter aby zakończyć konfigurację.

[15] ftp://ftp.univie.ac.at/packages/network/pegasus/

Dalej już będzie łatwo. W menu znajdziemy funkcję **Get new mail by POP3** używaną do sprowadzania poczty z serwera. Po jej przeczytaniu i przygotowaniu wszystkich listów wybieramy funkcję **Send a message** i dalej **Send all messages** aby wysłać wszystjkie listy.

10.3 NUPop

Poszukując najlepszego programu do obsługi poczty elektronicznej, już kilka lat temu „przetestowałwm" program **NUPop**. Dopiero jednak niedawno dowiedziałem się o jego ciekawych możliwościach: odczytu poczty z domu (z wykorzystaniem modemu) ale bez potrzeby (czy konieczności) instalowania i uruchamiania protokołu SLIP czy PPP!

Dziś coraz łatwiej (można powiedzieć, że jest to pewnym już standardem) o dostęp do sieci Internet z domu za pośrednictwem modemu i linii telefonicznej. Jest też takim standardem używanie protokołów SLIP lub PPP. Nie wszyscy jednak chcą (lub mogą) z tych udogodnień korzystać. Pozostaje im wówczas korzystanie w domu z połączenia terminalowego z serwerem UNIXowym za pośrednictwem programów takich jak **kermit** (patrz str. 11), **procomm**, **deputy**[16] czy dziesiątki innych. Na serwerze trzeba korzystać z programów pocztowych **pine**, **elm** czy (o zgrozo[17]) z programów **mail/mailx**.

Cóż to jest zatem **NUPop**?

Jest to program pozwalający wykorzystać komputer PC pracujący pod kontrolą systemu operacyjnego DOS do odczytu i wysyłania poczty elektronicznej.

Co jest potrzebne do wykorzystania programu?

1. Komputer PC; program NUPop dostępny jest w dwu wersjach:
   - *real mode* o minimalnych wymaganiach: procesor 8086/8088 i co najmniej 490k wolnej pamięci RAM (po załadowaniu wszystkich niezbędnych programów rezydentnych),
   - *protected mode* wymagający procesora 80386 lub lepszego i minimum 2 MB pamięci RAM; powinno być dostępne DPMI[18]

   Program wymaga ok 1 MB pamięci dyskowej.
2. Dostępu do serwera poczty POP. Na serwerze, na którym odbieramy pocztę musi być zainstalowany serwer POP.
3. Dostęp do sieci. Może być on zrealizowany w jeden z poniższych sposobów:
   - komputer wyposażony jest w kartę sieciową i pracuje w sieci lokalnej podłączonej do sieci Internet,
   - komputer wyposażony jest w modem
     - i dostępny jest protokół SLIP/PPP,
     - i mamy dostęp do serwera terminali, serwera komunikacyjnego lub konat na komputerze (na przykład UNIXowym), z którego możemy wydać polecenie **telnet**.
4. Odpowiednie oprogramowanie sieciowe (gdy komputer pracuje w sieci). Program **NUPop** może współpracować z następującym oprogramowaniem sieciowym:
   - *packet driver* (tak współpracujący z „klasyczną" kartą sieciową lub modemem i protokołami SLIP/PPP),
   - Novell Netware (potrzebny jest program **odipkt**),
   - Lan Manager (potrzebny jest program **ndis_pkt**),
   - LocalTalk (z wykorzystaniem drivera **localtalk**),
   - program poprawnie rozpoznaje i potrafi skorzystać z zainstalowanego jądra sieciowego komercyjnego pakietu PC/TCP firmy FTP Software Inc.
   - gdy korzystamy z oprogramowania sieciowego Lan Workplace for DOS, w zestawie dostępny jest program **nutcplwp.exe** zapewniający współprace z tym oprogramowniem sieciowym,
   - podobny program (o nazwie **nutcpwol.exe**) zapewnia współpracę z oprogramowaniem Wollongong PathWay,

[16] Można ten program polecić wszystkim tym, którzy nie posiadają modemu z kompresją i korekcją błędów – program załatwia to za modem.

[17] Spotkałem jednak człowieka, który odrzuca wszystkie „pełnoekranowe" narzędzia i korzysta z innych, typu **mush** lub **mail**.

[18] *DOS Protected Mode Interface.*

- program **nutcptrm.exe** zapewnia współpracę z DOSowym jądrem sieciowym napisanym przez Petera Tattama (dziś bardziej znanego jako autora drajwera winsock w środowisku Windows).

Aby zainstalować oprogramowanie potrzebować będziemy jeszcze pewne dodatkowe informacje:

- numer IP, adres komputera PC i maskę sieciową (*netmask*) (w przypadku pracy w sieci),
- adres serwera poczty (POP – odbiór poczty i SMTP – wysyłanie),
- adres routera łączącego sieć lokalną ze „światem”,
- adres serwera DNS i nazwę domeny, w której znajduje się nasz komputer.

Zdobyć musimy też oprogramowanie. Jest to niestety dosyć stary już program i nie będzie łatwo znaleźć „najnowszą” jego wersje. Polecam sięgnięcie do źrodeł[19] lub skorzystanie z archiego.

Powinniśmy wybrać wersję odpowiednią do używanego komputera. Można również skorzystać z dostępnej (w języku angielskim) dokumentacji.

Tworzymy kartotekę, w której znajdzie się program i wszystkie pliki konfiguracyjne. Rozpakowywujemy w niej archiwa. Po pierwszym uruchomieniu programu odpowiedzieć będziemy musieli na szereg pytań określających konfigurację systemu i preferencje użytkownika.

Nie będziemy tu omawiali znaczenia poszczególnych pytań (program wyposażony jest w stosunkowo bogaty system wypowiedzi). Poniżej zajmiemy się dodatkowymi czynnościami konfiguracyjnymi niezbędnymi do pracy w sytuacji gdy nie mamy dostępu do sieci – wykorzystujemy modem i konto na komputerze UNIXowym. Musimy przygotować specjalne „skrypty” (pliki w szczegółowy sposób określające czynności jakie powinien wykonać program w naszym imieniu aby automatycznie nawiązać połączenie z serwerem odebrać lub wyekspediować pocztę).

Aby poprawnie przygotować skrypt musimy bardzo dokładnie przyjrzeć się prcedurze nawiązywania połączenia z serwerem. Najlepiej podczas którejś z sesji włączyć opcję zapisywania do pliku i przeanalizować ten plik. W „moim” przypadku wyglądała ona tak:

```
ATD202784

CARRIER 14400

PROTOCOL: LAP-M

COMPRESSION: V.42BIS

CONNECT 14400/ARQ
```

Połączenie między modemami zostało nawiązane. W tym miejscu należy nacisnąć klawisz Enter dwukrotnie w krótkim odstępie czasu. Pozwala to zorientować się komputerowi, z którym jesteśmy połączeni z jaką szybkościa pracujemy. W każdym przypadku dalsze postępowanie będzie już różne.

Po dwukrotnym naciśnięciu klawisza Enter nawiązaliśmy połączenie.

```
PAD2002 Version 1.42 rev 4 (c) Netcomm Limited 1987-1990
NRS Lookup Protocol Version 1.0
(C) Netcomm Research Limited 1988-1990


DFN-PAD>
```

Po wybraniu numeru uzyskuję połączenie z węzłem sieci X.25, który pozwala na uzyskanie połączenia z jednym z wybranychj komputerów sieci WASK po wydaniu polecenia `call` *xxxx*, gdzie *xxxx* to numer komputera. Aby uzyskać połączenie z komputerem ldhpux.immt.pwr.wroc.pl należy podac numer*2772081*.

```
DFN-PAD>call 2772081
Calling.
Accepted.
Trying 156.17.8.1 ... Open
```

[19] ftp://ftp.cos.edu/pub/pub/nupop

```
Witamy z Zakladzie Dynamiki


login:
```

W tym miejscu należy się przedstawić:

```
login: myszka
Password:
```

A tu podać hasło (nie będzie ono wyświetlane):

```
Password:
Please wait...checking for disk quotas
You have mail.
TERM = (vt100)
```

System prosi o potwierdzenie typu terminala, proponując jako wartość domyślną **vt100**; naciskamy tylko Enter

```
(hp)[1024]myszka>
```

Pojawia się „*prompt*" znak zachęty – proces nawiązywania połączenia możemy uznać za zakończony.

„Skrypt" (a właściwie dwa), które będziemy przygotowywali mają zachowywać się tak jak człowiek przed klawiaturą. Pierwszy z nich powinien obejmować wszystkie te czynności, które muszą być wykonane, aby nawiązać połączenie z systemem. Drugi czynności pozwalające nawiązać połączenie TCP/IP ze wskazanym portem serwera poczty/POP.

Pierwszy z nich może wyglądać tak (linie rozpoczynające się od znaku gwiazdki „*" traktowane są jak komentarze):

```
*
*     ,,Restart'' modemu
*
modeminit
stext "~ATX3&C1&D2M0S37=11N1&K3\\K5&Q5|"
*
*       komenda stext wysyła zadeklarowany ciąg znaków "do modemu"
*       znak ~ (tylda) nakazuje odczekać 1 sekundę przed rozpoczęciem
*              wysyłania
*       znak | (pionowa kreska) oznacza wysłanie znaku końca linii
*
waitstring "OK"
*
*       komenda waitstring wskazuje na tekst odpowiedzi, którą
*       powinniśmy uzyskać, żeby móc kontynuować
*
stext "~ATS36=7S37=11S38=1S46=138S48=7|"
waitstring "OK"
*
*       Powyższe ciągi znaków zainicjowały używany przeze mnie modem
*
*       Teraz trzeba wybrać numer, operacja będzie powtarzana wskazaną
*       liczbę razy (w tym przypadku 10)
*
dial 10
*
*       Oczekujemy na uzyskanie od modemu odpowiedzi "CONNECT"
*       każde "delay" oznacza odczekanie 10 sekund
*
delay
delay
delay
```

```
*
*       Teraz wysyłamy dwa znaki CR, żeby ,,obudzić'' serwer
*       W odpowiedzi powinniśmy uzyskać ciąg znaków ,,DFN-PAD>''
*
repsendwait 10 5 "|~|" "DFN-PAD>"
*
delay
*
stext "~call 2772081|"
waitstring "login: "
*
*       ,,Wykręciliśmy'' numer komputera i oczekujemy na jego zgłoszenie
*
stext "~myszka|"
waitstring "Password:"
stext "~hasło|"
*
*       Przedstawiamy się i musimy podać hasło. Niestety hasło musi być
*       zapisane ,,odkrytym'' tekstem!
*
waitstring "(vt100) "
stext "~dumb|"
*
*       Następnie komputer odpytuje o typ terminala: odpowiedź ,,dumb''
*       ma zapobiec interpretacji wysyłanych sekwencji znaków
*
waitstring "myszka>"
*
*       Odpowiedź ,,myszka>'' jest (w moim przypadku) standardowym
*       znakiem ,,zachęty'' (prompt) - możemy uznać, że połączenie
*       zostało nawiązane
```

Drugi skrypt jest już bardzo prosty: zawiera komendę jaką należy wydać, żeby nawiązać połączenie ze wskazanym portem serwera. W przypadku komputera UNIXowego będzie to komenda **telnet**. Skrypt wygląda tak:

```
*
*       Nawiązanie połączenia z komputerem (%n oznacza nazwę serwera,
*       %p - numer portu)
*       Korzystamy ze ,,standardowego'' programu telnet - nawiązanie
*       połączenia sygnalizowane jest informacją ,,escape character is ^]''
*
stext "~telnet %n %p|"
*
waitstring "is '^]'."
*
```

I sprawa załatwiona! Zwracam uwagę, że nie należy „niewolniczo" kopiować powyższych skryptów. Należy dokładnie przyjrzeć się kolejności wykonywanych czynności i przetworzyć je na kilka prostych komend. Język używany przez program jest nieco bardziej rozbudowany – pozwala nawet na warunkowe wykonywanie komend.

10.4 Opis programu `elm` – jeszcze nie gotowe

I nie wiem czy znajdę czas na to :-(Może ktoś zna opis, który można tu powołac? Proszę o informacje.

Jest też dostępna w księgarniach książka Anny Trykozko i Michała Jarskiego [8]. Niestety nie czytalem i nie moge powiedzieć czy jest warta kupienia.

Jerzy Kędra zwrócił mi uwagę, że istnieje coś na kształt strony domowej[20] programu **elm**. Zapraszam do przeglądania, choć wszytskie informacje są po angielsku.

[20] http://www.myxa.com/elm.html

10.5 Opis programu Eudora

Nie korzystam z tego, ponoć niezłego programu obsługi poczty elektronicznej, pracującego w środowisku MS–Windows. Znalazłem jego opis i za zgodą autora (Lesława Sieniawskiego) zamieszczam tu odsyłacz. **Uwaga:** materiał jest bogato ilustrowany!

Rozdział 11

Kermit – jeszcze nie gotowe

Zwięzły opis programu **kermit**[1] może przydać się tym, którzy korzystają z dostępu do sieci wydzwaniając z domu i dalej pracują korzystając z konta na serwerze. Kermit jest programem oferującym kilka interesujących usług:

- bardzo dobrą emulacje terminala w środowisku DOS (program pracuje w środowisku MS Windows, ale **nie** jest to aplikacja okienkowa):
  - ansi
  - vt52
  - vt100, vt102
  - vt220
  - vt320
  - Wyse 50
  - honeywell
  - Tektronix 4010 (terminal graficzny)

  (i jeszcze kilku, rzadziej u nas spotykanych),
- bardzo efektywny transfer plików z serwara na komputer osobisty,
- przeźroczystą (w locie) translację znaków ze standardu ISO 8859–2 do CP 852.

Żeby było ciekawiej, program może wypełniać funkcje emulatora terminala w środowisku sieci lokalnej (TCP/IP).

Najważniejszą cechą tego programu jest jego cena: jest darmowy! Jedynie wersja pracująca w środowisku Windows 95—indexWindows 95 jest **komercyjna**.

Program został opracowany na Uniwersytecie Columbia (Nowy Jork) i najnowsza wersja pracująca w środowisku DOS nazywa sie MS–DOS Kermit i ma numer 3.14. Wersja 3.15 jest w ostatniej fazie testów beta. Wersja pracująca w środowisku UNIX (C–Kermit) i ma numer 6A(193).

Nieco więcej informacji na temat takiego skonfigurowania programu żeby poprawnie obsługiwał polskie litery znaleźć można w osobnym dokumencie.[2]

[1] Copyright (C) 1982, 1995, Trustees of Columbia University in the City of New York.
[2] ./dos/

Część II

Podręcznik administratora

Rozdział 12

named

Do normalnej pracy systemu komputerowego nie jest niezbędne uruchamianie programu named – do tego celu wystarczy poprawnie skonfigurowany **resolver** (patrz str. 5.3).

named jest specjalnym programem – serwerem odpowiadającym na zapytania **resolverów** dotyczące translacji nazw symbolicznych na adresy IP i odwrotnych. **named** podczas udzielania odpowiedzi kontaktuje się z innymi serwerami, a wszystkie uzyskane odpowiedzi przechowuje w tzw. „pamięci podręcznej" (*cache*). (Taka strategia postępowania pozwala uprościć proces odpowiedzi na kolejne pytania.)

Kiedy **musimy** uruchomić program **named**? Są trzy sytuacje w których musimy na komputerze uruchomić oprogramowanie:

1. Chcemy tego. Z jakichś przyczyn uznajemy, że będzie pożądane przechowywanie informacji o najczęściej wykorzystywanych adresach w pamięci jednego z komputerów podłączonych do sieci lokalnej.

   Z taką sytuacją możemy miec do czynienia wtedy, gdy połączenie sieci lokalnej z resztą Internetu realizowane jest po bardzo/stosunkowo wolnym łączu. Konfigurujemy wówczas jeden komputer w sieci lokalnej jako nameserwer i wskazujemy go jako podstawowe źródło informacji resolverom pozostałych komputerów. W ten sposób tworzymy *caching–only* serwer.

   Zaletą takiego rozwiązania jest to, że po jakimś czasie na lakalnym serwerze mamy większość informacji, z których korzystają lokalni użytkownicy. Oni dostają odpowiedzi na swoje zapytania szybko, wolne łącze jest odciążenie od wielokrotnego przekazywania tych samych informacji.

2. Jesteśmy odpowiedzialni za zarządzanie domeną adresową pewnej instytucji i nie chcemy (albo nie możemy) zlecić tej „fuchy" komu innemu.

   Musimy wówczas utworzyć główny (pierwotny ??) (ang. *primary*) serwer zawierające dane o domenie.

3. Administrator domeny poprosił/zażądał aby jeden z naszych komputerów spełniał rolę serwera zapasowego dla pewnej domeny.

   Wówczas tworzymy serwer wtórny (ang. *secondary*).

Oczywiście, prawie nigdy nie będziemy mieli do czynienia z żadną z powyższych sytuacji w czystej postaci. Możliwa jest praktycznie dowolna kombinacja ich wszystkich.

Ten sam komputer może równocześnie spełniać rolę serwera pierwotnego dla pewnych domen, wtórnego dla innych.

Omówimy poniżej pewne elementarne sprawy związane z uruchomieniem nameserwera (programu named) na komputerze UNIXowym.

Oprogramowanie jest najczęściej tak skonfigurowane, że nie trzeba dokonywać wielkich zmian konfiguracyjnych. W prostszych przypadkach należy jedynie utworzyć kilka plików i wypełnić je poprawnymi danymi.

12.1 Czynności przygotowawcze

Podstawowym plikiem czytanym przez program **named** jest plik `/etc/named.boot` zawierający niezbędne informacje:

- o nazwie kartoteki gdzie są umieszczone pliki „baz danych",
- adresy (czy raczej nazwa pliku je zwierającego) komputerów odpowiedzialnych za domenę główną,[1]

[1] Ze względu na hierarchiczną strukturę bazy danych – adresy te są minimalną informacją niezbędną do normalnej pracy nameserwera.

- informację o nazwach domen dla których nameserwer jest serwerem pierwotnym,
- informacje o nazwach domen dla których nameserwer jest serwerem wtórnym,
- nazwy serwerów które będą wykorzystywane do „zdobywania” wszelkich informacji (nasz będzie je tylko przechowywał).

Plik ten musi istnieć w chwili uruchamiania programu **named**, jego zawartość określa typ serwera z jakim mamy do czynienia.

12.1.1 Instalacje serwera „caching–only”

Każdy serwer (jeżeli już zdecydujemy się go uruchamiać) spełnia rolę serwera pierwotnego dla domeny 0.0.127.in-addr.arpa. Jak wiadomo przyjęto taką konwencję, że adres 127.0.0.1 jest to adres przypisany „sztucznemu” interfejsowi sieciowemu. Istnieje on niezależnie od podłączenia komputera do sieci. Umożliwia **lokalną** pracę oprogramowania sieciowego nawet na komputerach nie podłączonych do niej fizycznie. Z tego samego względu zastrzeżona nazwa **localhost** jest na każdym komputerze nazwą symboliczną tego interfejsu.

Plik `/etc/named.boot` w przypadku takiego serwera powinien zawierać:

- nazwę kartoteki, w której są przechowywane wszystkie dane,
- nazwę pliku zawierającego informacje o domenie głównej

i może wyglądać, na przykład tak:

```
directory                               /var/named
primary         0.0.127.in-addr.arpa    named.local
cache                                   named.ca
```

(Nazwa kartoteki, oraz nazwy wszystkich plików są zupełnie dowolne – i po to jest właśnie plik /etc/named.boot żeby wskazać programowi **named** gdzie ma szukać informacji.)

O tym, że serwer jest *caching–only* decyduje **nieobecność** lini rozpoczynających się od `primary` (za wyjątkiem informacji o demenie 0.0.127.in-addr.arpa) oraz `secondary`.

Zawartość pozostałych plików zostanie opisana później.

12.1.2 Instalacja serwera podporządkowanego (*slave*)

Serwer podporządkowany (*slave*) jest specyficzną cechą nameserwera, ktory nie kontaktuje się z serwerami domeny głównej. W związku z tym nie jest w stanie samodzielnie odpowiadać na zapytania na które nie znajduje odpowiedzi na podstawie zawartości swojej pamięci cache. Aby zadeklarować tę cechę serwear umieszcza się wiersz

```
slave
```

w pliku `/etc/named.boot`. Oczywiście taka konfiguracja nie ma wielkiego sensu (chyba, że i tak pracujemy w sieci, lokalnej która jest odcięta od reszty świata).

Dyrektywa `slave` używana jest najczęściej razem z dyrektywą `forwarders` która instruuje nameserwer, żeby jeżeli nie potrafi odpowiedzieć na zapytania na podstawie zawartości swojej pamięci zapytanie odsyłał do wskazanych nameserwerów. Dopiero jeżeli te nie udzielają odpowiedzi (są na przykład niedostępne) – serwer podejmuje próbe samodzielnego (jeżeli nie użyto dyrektywy `slave`) zdobycia informacji.

Dyrektywy `forwarders` można spróbować użyć w celu stworzenia bogatej bazy adresów na kilku serwerach sieci lokalnej: wszystkie serwery sieci (na przykład) miejskiej kontaktują się z dwoma – trzema komputerami, które gromadzą wszystkie dane o adresach (i tylko one dokonują rekurencyjnych zapytań o nowe adresy).

Postać dyrektywy `forwarders` jest następująca:
`forwarders` *server–address server–address. . .*

Gdzie *server–address* sa numerami IP serwerów wyznaczonych do zdobywania wszystkich informacji.

Każdy nameserwer (caching–onlly, główny, wtórny. . .) może być skonfigurowany tak, żeby korzystać z pomocy innych nameserwerow lub jako serwer podporządkowany.

UWAGA: Przed wpisaniem cudzego nameserwera w lini `forwarders` należy spytać się administratora czy sobie tego życzy: Na słabszych komputerach **named** może zużywać znaczną część zasobów (zwłaszcza pamięci).

12.1.3 Instalacja serwera głównego (*primary*)

Serwer *primary*, oprócz informacji niezbędnych dla serwera caching–only wymaga dodatkowo plików opisujących domenę. Mogąto być pliki opisujące domenę „prostą" i/lub „odwrotną". Pik /etc/named.boot wyglądać może w sposób następujący:

```
directory                                /var/named
primary        0.0.127.in-addr.arpa      named.local
cache          .                         named.ca
primary        wroc.pl                   named.hosts
primary        17.156.in-addr.arpa       named.rev
```

Co oznacza, że w kartotece `/var/named` w pliku `named.hosts` zawarte są informacje zawierające powiązania pomiędzy nazwami symbolicznymi komputerów (węzłów) w domenie wroc.pl a ich numerami IP. W pliku `named.rev` zawarte są powiązania odwrotne: pomiędzy numerami IP a nazwami symbolicznymi.

12.1.4 Instalacja serwera wtórnego (*secondary*)

Zawartość pliku `/etc/named.boot` w tym przypadku jest następująca:

```
directory                                /var/named
primary        0.0.127.in-addr.arpa      named.local
cache          .                         named.ca
secondary      pwr.wroc.pl               156.17.5.2      pwr.wroc.pl.hosts
secondary      8.17.156.in-addr.arpa     156.17.8.1      8.17.156.rev
```

Tu widzimy pewne zmiany: rekord secondary zawiera cztery pola: po nazwie domeny, o której informacje przechowuje zawarty jest adres (mogą to być adresy) komputera który jest serwerem głównym dla tej domeny (i w związku z tym źródłem pewnych informacji o domenie) dodatkowe pole (**nieobowiązkowe**): nazwę pliku, gdzie będą zapisywane wszystkie informacje. Oczywiście wszelkie informacje, które zdobędzie **named** przechowywane są w pamięci operacyjnej komputera. Na dysku przechowywana może być jedynie kopia zapasowa (uzupełniana na bieżąco) tych informacji. Pozwala to zajrzeć do tego pliku, a również rozpocząć pracę serwerowi wtórnemu nawet wtedy, gdy serwer pierwotny jest niedostępny (nie ma połączenia sieciowego, komputer jest wyłączony, itp.).

12.2 Standardowe rekordy baz danych nameserwera

Wszystkie pliki opisujące bazy danych mają identyczną strukturę. Każdy z rekordów ma następującą budowę:
[name] [ttl] **IN** type data
gdzie:

name nazwa obiektu bazy danych, którego dotyczą wpisywane informacje; może być to pojedyńczy komputer lub cała domena; ciąg znaków traktowany jest jako nazwa względna (w stosunku do bieżącej domeny) – chyba, że zakończony jest kropką. Jeżeli pole jest puste – informacje dotyczą obiektu opisywanego w poprzednim rekordzie. Kropka (lub rzadziej używane dwie kropki) określają domenę główną. Pojedyńczy znak at („@") oznacza domenę „bieżącą" (nieco więcej o tym przy opisie rekordu SOA (patrz str. 12.2.1)). Gwiazdka („*") spełnia rolę „wzorca" (*wildcard*) i jest zgodna z dowolnym ciągiem znaków (nieco więcej będzie na ten temat przy opisie rekordu MX (patrz str. 12.2.4)).

ttl – *time to live* czas (wyrażony w sekundach) przez jaki dane powinny być przechowywane w pamięci cache nameserwerów (gdy już się tam znajdą). Gdy pole jest puste – przyjmuje się wartość określoną dla całej domeny w rekordzie typu SOA (patrz str. 12.2.1). Wielkość pola: co najwyżej 8 znaków; liczba 99999999 traktowana być może jako nieskończoność (więcej niż trzy lata).

IN informacja obowiązkowa, określa typ rekordu jako rekord DNS w Internecie (są również inne typy rekordów, ale nie mają one znaczenia w systemie DNS).

type określa rodzaj zasobów opisywanych przez rekord (patrz niżej).

data dane specyficzne dla każdego rodzaju rekordu (opisane zostaną w dalszej części).

Typy rekordów

SOA – Start of Authority: rekord ten oznacza początek danych dotyczących domeny; każda domena ma tylko jeden rekord SOA. Zwykla dane dotyczące domeny przchowywane są w osobnym pliku – stąd w każdym pliku jest tylko jeden taki rekord.

NS – Name Server: rekord wskazuje autorytatywny nameserwer dla nazwanej w polu name domeny. Są to podstawowe rekordy wiążące informacje ze wszystkich serwerów w hierarchiczną strukturę.

A – Address: wiąże nazwe z numerem IP; występuje w plikach opisujących powiązania „proste”.

MX – Mail Exchanger: taki rekord wskazuje komputery pełniące rolę serwera poczty dla całej domeny, lub przekaźnika poczty (komputera, który przejmuje i przechowywuje pocztę skierowaną na adres wskazany w polu name).

CNAME – Canonical Name: definiuje alias (nazwę zastępczą) komputera.

PTR – Pointer: rekord ten (odwrotnie do rekordu **A**) wiąże numer IP z nazwą symboliczną komputera; rekordy występują jedynie w opisie domen *.in-addr.arpa

HINFO – Host Information: pozwala zamieścić krótkie informacje na temat sprzętu i oprogramowania węzła.

WKS – Well Known Services (pisaliśmy już o tym na str. 3.2): rekord opisuje usługi świadczone przez węzeł. Rekord jest odzwierciedleniem zawartości pliku `/etc/services`. Dla każdego węzła może byc co najwyżej dwa takie rekordy: jeden dla usług świadczonych za pomocą TCP, drugi UDP. Rzadko stosowany.

TXT – pozwala wpisać dowolne informacje tekstowe opisujące domenę; w szczególności, można tu wpisać dane ułatwiające funkcjonowanie systemu **netfind**.

Oprócz wyżej wymienionych i powszechnie uznanych jest jeszcze parę typów rekordów, które niosą dodatkowe informacje, ale albo nie zostały jeszcze powszechnie zaimplementowane, albo nie znalazły szerszego zastosowania – stąd są używane niezmiernie rzadko.

12.2.1 Rekord SOA

Format rekordu SOA jest następujący:
[zone] [ttl] **IN SOA** *origin contact* **(**
serial
refresh
retry
expire
minimum
)

Znaczenie poszczególnych pól jest następujące:

zone nazwa domeny; Najczęściej w tym miejscu (rekord wpisujemy tylko na komputerze będącym nameserwerem głównym!) wpisuje się znak „@” (at) – tak użyty nawiązuje on do nazwy domeny użytej w dyrektywie **primary** znajdującej się w pliku named.boot a kierującej do pliku zawierającego rekord SOA.

ttl czas ważnowści rekordu – zawsze pozostawia się puste.

origin nazwa komputera będącego serwerem pierwotnym dla domeny; zwykle wpisywana jako pełna nazwa komputera (FQDN – *Fully Qualified Domain Name*).

contact adres osoby odpowiedzialnej za domenę. Adres jest zmodyfikowany w ten sposób, że znak „@” oddzielający nazwę użytkownika od nazwy komputera zastępuje się kropką. Zatem adres `hostmaster@immt.pwr.wroc.pl` zapisany będzie jako `hostmaster.immt.pwr.wroc.pl`. Najczęściej w przykładach znajdujących się w dokumentacji nazwa użytkownika to hostmaster. Proszę nie zapominać o powiązaniu tego adresu z adresem fizycznej osoby czytającej (i odpowiadającej) na pocztę. **Można tu wpisać adres dowolnej osoby. Nie musi on znajdować się na komputerze wymienionym w polu *origin*.**

serial numer wersji danych zawartych w pliku. Powinna to być liczba całokwita (niektóre, stare wersje programu named mają kłopoty z liczbami zawierającymi kropkę dziesiętną). Trudno zalecić jakąś zasadę numerowania wersji: doświadczenie uczy, że jeżeli wybierzemy jakąś zasadę – powinniśmy się jej trzymać. Zwłaszcza trzeba pamiętać, żeby numery wersji **rosły**. Znam przypadek, gdy jeden z administratorów zmienił zasady numeracji - rozpoczął ją w pewnym momencie od zera. Oczywiście znalazł się w sieci komputer, który był serwerem wtórnym tej domeny (nie został co prawda wyznaczony do tej roli). W plikach miał stary numer wersji i nie przyjmował nowej zawartości plików o niższych numerach wersji.

Bardzo dobrym zwyczajem jest numerowanie kolejnych wersji datą: rrmmddv[2] (jednocyfrove pole v pozwala na odnotowywanie drobnych zmian robionych jednego dnia), na przykład 1995050101.

refresh określa czas (w sekundach) co jaki serwery wtórne sprawdzają czy nie wzrósł numer wersji informacji pamiętanych na serwerze pierwotnym. Jeżeli numer zwiększył sie – serwer wtórny prosi o przetransferowanie zawartości całego pliku. Jeżeli nie nie podejmuje żadnych działań.

Serwer wtórny zawsze dokonuje takiego sprawdzenia w momencie startu i wtedy, kiedy odbierze sygnał SIGHUP.

Trudno zalecić jakąś konkretną wartość. Należy przyjąć zasadę, że jeżeli planujemy jakieś poważne zmiany i rekonfiguracje sieci, które będą miały istotny wpływ na zawartość baz danych – należy z odpowiednim wyprzedzeniem zmienić (zmniejszyć) ten okres (zmieniając oczywiście numer wersji na większy).

W większości przypadków odświeżanie raz na dobę jest (86400 s) jest wielkością poprawną.

retry określa jak długe serwer wtórny powinien czekać z powtórną próbą odświeżenia informacji o domenie, gdy poprzednia się nie powiodła. Czas ten nie powinien być zbyt krótki – gdyż zbytnio absorbuje zasoby serwerów wtórnych oraz zasoby sieciowe. Dobrą wartością jest 3600 (godzina) czy 1800 (pół godziny).

expire liczba ta określa po jakim czasie nieodświeżone dane o domenie poeinny być skasowane z pamięci serwerów wtórnych. Zwykle jest to bardzo duży czas, na przykład 3600000 (ok. 42 dni).

minimum liczba określa domniemaną wartość czasu życia (*ttl*) wszystkich rekordów – o ile nie zadeklarowano inaczej. Dobierając jej wartość należy uwzględniać jeden tylko fakt: jak długo umieszczone w bazie danych informacje będą aktualne. Jeżeli nie planujemy żadnych zmian – wartość może być bardzo duża: 1 tydzień (604800), a nawet 1 miesiąc (2592200). Zwracam jednak uwagę, że przez taki okres dane moga "leżeć" na odleglym serwerze (nie wszystkie komputery na swiecie restartuje się co dwa dni) i jezeli zechcemy dokonać jakichś zmian (nazwa komputera, numer IP) to będziemy w kłopocie. Dlatego też wszelkie istotne zmiany powinniśmy poprzedzić (najlepiej z wyprzedzeniem większym niż *ttl*) zmianą ttl na niewielka wartość (doba, kilka godzin) i dopiero robić zmiany. Nie ma potrzeby dokonywać takiej zmiany dla całej domeny – można to zrobić dla pojedyńczych rekordów.

Oczywiście długi czas życia rekordów w pamięci *cache* ma również pewien minus – obciąża pamięć zdalnych serwerów.

Przykładowy rekord SOA:

```
@   IN   SOA  ldhpux.immt.pwr.wroc.pl. hostmaster.ldhpux.immt.pwr.wroc.pl. (
                1995050502 ; Serial
                28800      ; Refresh (28800)
                7200       ; Retry
                604800     ; Expire
                86400      ; Minimum TTL 1 day
                )
```

NASK (w chwili obecnej zarządca domeny **.pl**) określił „typowe" wartości dla parametrów rekordu SOA. Można znaleźć je na ich serwerze ftp.[3] Tu zacytuję jedynie podstawowe informacje:

- Gdy domenę obsługuje tylko primary nameserver, względnie gdy primary i secondary nameservery są w tej samej (pod)sieci:

  Refresh time 10800 (3 godz.)

[2] Jeszcze lepszym rozwiązaniem może być konwencja rrmmddvv lub nawet rrrrmmddvv.
[3] ftp://ftp.nask.pl/pub/dns/dns-pl.txt

Retry time 3600 (1 godz.)

Expire time 604800 (7 dni)

Default TTL 86400 (1 dzień)

- Gdy domenę obsługuje dwa lub więcej nameserverów, przy czym chociaż jeden z nich jest w innej (pod)sieci:

  Refresh time 21600 (6 godz.)

  Retry time 7200 (2 godz.)

  Expire time 1209600 (14 dni)

  Default TTL 172800 (2 dni)

- Powyższe wartości mają charakter minimalnych, tzn. można je zwiększać w pewnych rozsądnych granicach. Zalecane jest jednak nieprzekraczanie poniższych maksymalnych wartości tych parametrów czasowych:

  Refresh time 86400 (1 dzień)

  Retry time 3600 (3 godz.)

  Expire time 2592000 (30 dni)

  Default TTL 604800 (7 dni)

12.2.2 Rekord NS

Rekord ten definiuje serwery, które posiadają autorytatywne informacje o domenie. To właśnie te rekordy stanowią łączniki budujące strukturę nameserwisu. Bez rekordów definiujących serwery dla „poddomeny" nie bedą dostępne żadne o niej informacje. Patrz również informacje o delegacji poddomeny (str. 12.4).

Format rekordu jest nastepujący:
[domain] [ttl] **IN NS** *server*

domain nazwa domeny, dla której w polu *server* definiujemy serwer autorytatywny.

ttl czas życia (zwykle puste).

server nazwa symboliczna serwera; żada się, żeby dla każdej domeny były zdefiniowane co najmniej dwa serwery (w tym jeden poza domeną). Aby nie było nieporozumień nazwę powinno się wpisywać jako FQDN[4] nawet jeżeli definiuje komputer wewnątrz domeny.

12.2.3 Rekord A

Sa to podstawowe rekordy wiążące nazwy symboliczne z numerami IP. Umieszcza się je najczęściej w pliku `named.hosts` (patrz str. 12.3.3).

Postać rekordu jest następująca:
[host] [ttl] **IN A** *address*
gdzie:

host nazwa węzła, którego numer definiujemy. Zwykle podawana jako nazwa wzgledna.

ttl Czas życia rekordu. Najczęściej puste (patrz również dyskusję przy omawianiu rekordu SOA (str. 12.2.1))

address adres interfejsu komputera zapisany jako cztery liczby dziesiętne oddzielone kropkami, np 156.17.8.1.

W większości przypadków adresy dotyczą wezłów znajdujących się wewnątrz domeny. I jako generalną zasadę należy przyjąć unikanie definiowania adresów IP komputerów spoza domeny. Chodzi tu o unikanie takiej sytuacji gdy w plikach zarządzanych przez nas wpisujemy informacje, które mogą być zmieniąne bez naszej woli. Jest jeden wyjątek od tej zasady – będzie o tym mowa przy omawianiu delegacji doemny (str. 12.4).

[4]Fully Qualified Domain Name – pełna nazwa zakończona kropką.

12.2.4 Rekord MX

Rekord wskazuje komputery, które mogą przejąć (czasowo lub na stałe) pocztę kierowaną na określony adres.

Postać rekordu jest następująca:
[name] [ttl] **IN MX** *preference host*
gdzie:

name nazwa komputera lub domeny – należy myśleć tu o tym co następuje po znaku „@”. Poczta wysyłana na wskazany adres będzie kierowana na adres wskazany w polu *host*.

ttl czas życia rekordu.

preference komputer lub domena może mieć więcej niż jeden komputer wskazany jako odbiorca poczty, wpisana w tym polu liczba okresla preferencję – im mniejsza jej wartość tym priorytet wyższy.

host nazwa serwera pocztowego.

Głównym zastosowaniem rekordu MX jest kierowanie poczty do komputerów (adresow pocztowych), które albo nie są podłączone bezpośrednio do Internetu, albo nie zostało na nich uruchomione oprogramowanie pocztowe. Zakłada się, że wskazany w rekordzie komputer przejmie pocztę i „coś” z nią zrobi (dostarczy odbiorcom w jakiś sposób).

Drugim zastosowaniem rekordu MX jest wskazanie komputerów, które przejmą chwilowo pocztę (gdy jej odbiorca jest niedostępny lub trudno dostępny). Tak jak pokazuje to poniższy rysunek:

```
  +-------+                                    +-------+
  |       |            wolne lacze             |       |
  |       |------------------------------------|       |
  |       |                                    |       |
  +-------+                                    +-------+
komputer ,,MX''                          komputer ,,docelowy''
```

Idea tego rozwiązania jest taka: definiujemy dla komputera „docelowego” co najmniej dwa rekordy MX; jeden (o wyższym priorytecie – na przykład 0) wskazuje na komputer docelowy, drugi (o niższym priorytecie) na komputer „MX”. Gdy wolne (zawodne łącze) jest nieczynne, zapchane lub komputer docelowy wyłączony – poczta dotrze na komputer „MX” i zostanie tam przechowana przez jakiś czas (oprogramowanie pocztowe będzie próbowało co jakiś czas pocztę dostarczyć na adres docelowy). Oczywiście powinniśmy spytać o zezwolenie administratorów komputerów, które zamierzamy wykorzystać w tym celu.

Z pewną ostrożnością można używać w polu *name* znaku „*” (gwiazdka). Trzeba jednak dobrze rozumieć zasadę działania programów wysyłających pocztę. Jest ona taka:

- Wysyłamy list na adres user@cos.cos.cos.
- Program wysyłający pocztę pyta się:
  - w pierwszej kolejności o rekord MX dla adresu cos.cos.cos – jeżeli znajdzie – używa tego adresu jako adresu odbiorcy poczty,
  - w następnej kolejności o rekord A dla adresu cos.cos.cos
- gdy to nie powiedzie się brane są pod uwagę rekordy MX dla
  - *.cos.cos
  - *.cos

Z powyższego postępowania wynikają dwa wnioski:

1. Warto dla każdego komputera, który będzie odbiorcą poczty zadeklarować rekord MX (wskazujący na niego),
2. Rekordy typu:

   ```
   *.pl    IN      MX  10   host
   ```

   mające w zamiarze kierować całą pocztę do domeny `pl` przez komputer o nazwie `host` nie przyniosą żadnego efektu.

Można natomiast próbować wstawiać rekord następującego rodzaju:

```
*.bitnet.         IN      MX  0   plearn.edu.pl.
```

kierujący całą pocztę na adresy bitnetowe na komputer plearn.edu.pl (znany również jako plearn.bitnet) bez dokonywania przeróbek w pliku `sendmail.cf`.

12.2.5 Rekord CNAME

Rekord ten pozwala deklarować kilka nazw alternatywnych dla każdego komputera. Pozwala to na wygodne manipulowanie powiązaniem funkcji komputerów (związanych z łatwym do zapamiętania adresem) z fizyczną ich lokalizacją. Na przykład próbujemy swoich sił w stawianiu serwera WWW. Wybieramy do tego celu jakiś najmniej obciążony komputer, nadajemy mu alias „www.domena", uruchamiamy serwer, wypełniamy go wstępnymi informacjami i ogłaszamy o jego istnieniu.

Ponieważ mielismy znakomity pomysł na serwowane informacje – rozpoczyna się duży ruch, komputer jest za słaby. Wybieramy zatem inny, mocniejszy. Instalujemy tam serwer i wszystkie informacje. Dokonujemy zmiany aliasa. I tak, w niezauważony dla nikogo sposób możemy dokonać przeniesienia funkcji z komputera na komputer.

Podobne postępowanie może się przydać w przypadku gdy dotyczczasowa nazwa komputera nam się znudziła, lub z jakichś powodów musieliśmy wyłączyć komputer sprawujący ważne funkcje i zastąpić go innym.

Format rekordu jest następujący:
[nickname] [ttl] **IN CNAME** *host*
Gdzie:

nickname alias (pseudonim, dodatkowa nazwa).

ttl czas życia rekordu.

host oficjalna nazwa komputera.

Oto fragment informacji opisujących komputer ldhpux.immt.pwr.wroc.pl pełniący również rolę serwera ftp:

```
ldhpux  IN      A       156.17.8.1
ftp     IN      CNAME   ldhpux
```

i występujący również pod nazwą ftp.immt.pwr.wroc.pl.

12.2.6 Rekord PTR

Rekord **PTR** spełnia w plikach tzw. baz danych odwrotnych analogiczną rolę do rekordu **A** – łączy numery IP z adresami symbolicznymi. Służą do konstrukcji hierarchii domen *in-addr.arpa*.

Zdarza się czasami, że administratorzy nie przywiązują zbytniej wagi do prawidłowości danych w domenie odwrotnej. Dane zawarte w tej bazie danych nie mają bowiem istonego wpływu na działanie wielu komend. Zwracam jednak uwagę, że coraz więcej komputerów odmawia świadczenia usług (lub świadczy je jedynie w ograniczonym zakresie) gdy zamiana numeru IP na nazwę symboliczną i translacja odwrotna (numer na nazwę) nie dają tego samego rezultatu.

Format rekordu jest następujący:
name [ttl] **IN PTR** *host*

name powinno być właściwie napisane *numer* **:-)** jest on określony względnie aktualnej domeny; to znaczy jeżeli aktualna domena to 17.156.in-addr.arpa, a komputer ldhpux.immt.pwr.wroc.pl ma numer 156.17.8.1, wówczas pole *name* powinno zawierać „1.8".

ttl czas życia rekordu, najczęściej puste.

host pełna nazwa (FQDN – patrz również na str. 5.2).

12.2.7 Rekord HINFO

Rekord HINFO pozwala umieścić, krótkie informacje na temat sprzętu i oprogramowania komputera podłączonego do sieci. Zwracam uwagę, że odpowiednie dokumenty definiujące postać zawartych tam informacji wymagają, aby do określenia typu używanego sprzętu i oprogramowania używać zdefiniowanych w odpowiednim rfc1340[5] symboli.

Ponieważ wiedza na temat używanego oprogramowania może być potencjalnie przydatna dla włamywacza – wielu administratorów nie umieszcza tych informacji.

Postać rekordu jest następująca:
host [ttl] **IN HINFO** *hardware software*

[5] ftp://ftp.immt.pwr.wroc.pl/pub/rfc/1340.txt

host nazwa komputera, którego informacja ta dotyczy,

ttl czas życia (najczęściej puste),

hardware określenie sprzętu (jeżeli nazwa zawiera odstępy – powinna być ujęta w cudzysłowy),

software nazwa systemu operacyjnego.

12.2.8 Rekord WKS

Rekord ten pozwala na wyspecyfikowanie usług, które świadczy komputer. Nie jest zbyt często używane – ale powinno odzwierciedlać zawartość pliku `/etc/services`. Dla każdego komputera mogą być co najwyżej dwa rekordy: jednen – opisujący usługi dostępne za pośrednictwem protokołu TCP, drugi – UDP.

Postać rekordu jest następująca:
host [ttl] **IN WKS** *address protocol services*

host nazwa komputera, ktorego rekord dotyczy,

ttl czas życia rekordu,

address adres IP komputera zapisany jako cztery liczby dziesiętne oddzielone kropkmi,

protocol TCP lub UDP,

services lista usług oferowanych przez komputer; można wpisać tyle usług ile chcemy – ich nazwy powinny być zgodne z tymi wpisanymi do pliku `/etc/services`.

Bardzo często względy bezpieczeństwa komputera decydują, że rekord nie jest umieszczany w bazach danych.

12.2.9 Rekord TXT

Postać rekordu jest następująca:
name [ttl] **IN TXT** *ciąg znaków*
Znaczenie parametrów standardowe.

ciąg znaków dowolny tekst zawierające informacje, bardzo dobrze jest łańcuch znaków zamknąć w cudzysłowy („"”).

12.3 Pliki bazy danych nameserwera

Opiszemy tu najbardziej elementarne dane, które powinny znaleźć się w plikach bazy danych. Zwracam uwagę, że nie będzie to **kompletny** opis. W każdym pliku będą się mogły znaleźć dane w bogatszej konfiguracji – system jest bardzo elastyczny.

12.3.1 Zawartość pliku `named.local`

Plik `named.local` służy do powiązania interfejsu o numerze 127.0.0.1 z nazwą *localhost*. Raz jeszcze zwracam uwagę, że każdy komputer podłaczony do sieci powinien rozpoznawać nazwę *localhost* i interfejs o adresie *127.0.0.1* jako lokalny. Pozwala to na wykorzystywanie wszelkich mechanizmów komunikacyjnych na kompuetrze nie podłączonym do sieci.

Przykładowy plik moze wyglądać tak:

```
@       IN      SOA     ldhpux.immt.pwr.wroc.pl. hostmaster.ldhpux.immt.pwr.wroc.pl. (
                        1
                        36000
                        3600
                        3600000
                        36000
                        )
        IN      NS      ldhpux.immt.pwr.wroc.pl.
1       IN      PTR     localhost.
```

12.3.2 Zawartość pliku `named.ca`

Jest to podstawowy plik zawierający informacje o nameserwerach dla domeny głównej (*root*) oznaczanej również `.` (kropka). Zawiera on informacje, które są niezbędne do skonfigurowania nameserwera domeny prostej.

Można tam umieścić dodatkowe informacje (nameserwer będzie je znał zaraz po starcie). Nie zalecam jednak wpisywania tam informacji, ktore mogą się zbyt szybko zmieniać – zwykle zapominamy o tym pliku i prowadzi to do błędów i nieporozumień gdzy zmieni sie adres lub numer IP komputera który umieściliśmy w tym pliku.

Zwykle dokumentacja komputera oraz przykładowe pliki konfiguracyjne nameserwera zawierają plik, który może być wykorzystany. Do podanej tam informacji trzeba podchodzić z pewną dozą rezerwy. Najczęściej informacje tam zawarte są troche (czasami tylko nieco) przeterminowane.

Jak zatem zdobyć poprawne informacje:

1. Najpewniejsze informacje zawarte są u samego źródła i dostępne via ftp[6] lub gopher[7]
2. wpisać wartości z dokumentacji, a następnie używając programu **nslookup** (patrz str. 5.4) sprawdzić poprawność informacji:

   ```
   $ nslookup
   > set type=ns
   > .
   ```

   (ta „.” (kropka) to właśnie zapytanie o nameserwery dla domeny głównej (*root*)). Powinniśmy (jeżeli wszystko jest OK) otrzymać aktualną listę serwerów. (Oczywiście otrzymane dane należy nieco zmodyfikować, tak, żeby były zrozumiał dla programu **named**.)
3. Korzystając z programu **nslookup** (patrz str. 5.4) spytać się innego nameserwera (takiego, któremu wierzymy):

   ```
   $ nslookup
   > server ns.internic.net
   > set type=ns
   > .
   ```
4. Można wreszcie skorzystać z hipertekstowego formularza[8]
5. Z obowiązku zamieszczam listę aktualną w dniu pisania tego tekstu (6 sierpnia 1995):

   ```
   .                        99999999      NS      A.ROOT-SERVERS.NET
   .                        99999999      NS      B.ROOT-SERVERS.NET
   .                        99999999      NS      C.ROOT-SERVERS.NET
   .                        99999999      NS      D.ROOT-SERVERS.NET
   .                        99999999      NS      E.ROOT-SERVERS.NET
   .                        99999999      NS      F.ROOT-SERVERS.NET
   .                        99999999      NS      G.ROOT-SERVERS.NET
   .                        99999999      NS      H.ROOT-SERVERS.NET
   .                        99999999      NS      I.ROOT-SERVERS.NET

   A.ROOT-SERVERS.NET       99999999      A       198.41.0.4
   B.ROOT-SERVERS.NET       99999999      A       128.9.0.107
   C.ROOT-SERVERS.NET       99999999      A       192.33.4.12
   D.ROOT-SERVERS.NET       99999999      A       128.8.10.90
   E.ROOT-SERVERS.NET       99999999      A       192.203.230.10
   F.ROOT-SERVERS.NET       99999999      A       39.13.229.241
   G.ROOT-SERVERS.NET       99999999      A       192.112.36.4
   H.ROOT-SERVERS.NET       99999999      A       128.63.2.53
   I.ROOT-SERVERS.NET       99999999      A       192.36.148.17
   ```

W każdym z powyższych przypadków można się upewnić czy posiadane dane są prawidłowe. W tym celu trzeba wybrać dowolny z serwerów i za pomocą programu **nslookup** jego zapytać (w sposób analogiczny do podanego w pkt. 2) o serwery.

[6] ftp://FTP.RS.INTERNIC.NET/domain/named.root
[7] gopher://rs.internic.net/
[8] http://www.immt.pwr.wroc.pl/htbin/nsl

12.3.3 Zawartosc pliku `named.hosts`

Plik ten zawiera podstawowe informacje o domenie i wszystkich komputerach w niej się znajdujących. Służy on do zamiany nazw komputerów na numery IP (rekordy **A** (patrz str. 12.2.3)), zawiera informacje aliasach nazw (rekordy **CNAME** (str. 12.2.5)), odbiorcach poczty (rekord **MX** (str. 12.2.4)). W tym pliku będą też zawarte informacje o serwerach „poddomen” – jeżeli struktura jest rozbudowana (rekord **NS** (str. 12.2.2)). Natomiast w opisie **obowiązkowo** powinny się znaleźć informacje o adresach wszystkich oficjalnych name-serwerów domeny.

Niektórzy umieszczają tu również informacje o sprzęcie (rekord **HINFO** (str. 12.2.7)), usługach swiadczonych przez komputery (rekordy **WKS** (str. 12.2.8)). Można również umieścić rekordy **TXT** (str. 12.2.9) aby zawrzeć dodatkowe dane o domenie – pozwalające na przekazanie pewnych informacji dla systemu adresowego X.500.

Przykładowy (rzeczywisty) plik opisyjący pewną domenę zamieszczamy poniżej:

```
@ IN SOA ldhpux.immt.pwr.wroc.pl. hostmaster.ldhpux.immt.pwr.wroc.pl. (
                1996021302 ; Serial
28800      ; Refresh (28800)
7200       ; Retry
604800     ; Expire
86400      ; Minimum TTL 1 day
                     )
IN NS ldhpux.immt.pwr.wroc.pl.
IN NS sun2.pwr.wroc.pl.
localhost IN A 127.0.0.1
lx IN A 156.17.14.1
IN MX 0 lx.ikem.pwr.wroc.pl.
IN MX 10 sun1000.pwr.wroc.pl.
cadmes IN A 156.17.14.2
IN MX 0 cadmes.ikem.pwr.wroc.pl.
IN MX 10 sun1000.pwr.wroc.pl.
mech IN A 156.17.14.3
IN MX 0 mech.ikem.pwr.wroc.pl.
IN MX 10 sun1000.pwr.wroc.pl.
tmm IN A 156.17.14.5
IN MX 0 tmm.ikem.pwr.wroc.pl.
IN MX 10 sun1000.pwr.wroc.pl.
mirek IN A 156.17.14.19
IN MX 0 mirek.ikem.pwr.wroc.pl.
IN MX 10 sun1000.pwr.wroc.pl.

jacek IN A 156.17.14.13
tomi16 IN A 156.17.14.20
labcad IN A 156.17.14.21
grzyb01 IN A 156.17.14.12
grzyb02 IN A 156.17.14.14
grzyb03 IN A 156.17.14.15
grzyb04 IN A 156.17.14.16
grzyb05 IN A 156.17.14.17
grzyb06 IN A 156.17.14.18
grzyb07 IN A 156.17.14.8
grzyb08 IN A 156.17.14.22
grzyb09 IN A 156.17.14.9
grzyb10 IN A 156.17.14.24
lj4 IN A 156.17.14.7
```

12.3.4 Zawartosc pliku `named.rev`

W tym pliku zawarte są informację łączące numery IP z nazwami komputerów. Plik zawiera oprócz rekordu **SOA** i rekordów **NS** jedynie rekordy **PTR**. Plik bazy danych dla doemny opisanej w poprzednim rozdziale będzie wyglądał tak:

```
@ IN SOA ldhpux.immt.pwr.wroc.pl. hostmaster.ldhpux.immt.pwr.wroc.pl. (
              1996021301 ; Serial
10800     ; Refresh 24 hours
3600      ; Retry 2 hours
432000    ; Expire 30 days
86400     ; Minimum TTL 1 day
                   )
IN NS ldhpux.immt.pwr.wroc.pl.
IN NS sun2.pwr.wroc.pl.
1 IN PTR lx.ikem.pwr.wroc.pl.
2 IN PTR cadmes.ikem.pwr.wroc.pl.
3 IN PTR mech.ikem.pwr.wroc.pl.
5 IN PTR tmm.ikem.pwr.wroc.pl.
7 IN PTR lj4.ikem.pwr.wroc.pl.
8 IN PTR grzyb07.ikem.pwr.wroc.pl.
9 IN PTR grzyb09.ikem.pwr.wroc.pl.
12 IN PTR grzyb01.ikem.pwr.wroc.pl.
13 IN PTR jacek.ikem.pwr.wroc.pl.
14 IN PTR grzyb02.ikem.pwr.wroc.pl.
15 IN PTR grzyb03.ikem.pwr.wroc.pl.
16 IN PTR grzyb04.ikem.pwr.wroc.pl.
17 IN PTR grzyb05.ikem.pwr.wroc.pl.
18 IN PTR grzyb06.ikem.pwr.wroc.pl.
19 IN PTR mirek.ikem.pwr.wroc.pl.
20 IN PTR tomi16.ikem.pwr.wroc.pl.
21 IN PTR labcad.ikem.pwr.wroc.pl.
22 IN PTR grzyb08.ikem.pwr.wroc.pl.
24 IN PTR grzyb10.ikem.pwr.wroc.pl.
```

12.4 Delegacja domeny

Delegacja domeny jest czynnością stosunkowo prostą, i odbywa się zawsze według jednego schematu:

- Administrator domeny pewnego poziomu zostaje poproszony o utworzenie domeny o nazwie dłuższej o jeden czlon: administrator domeny `pwr.wroc.pl` proszony jest o utworzenie domeny `immt.pwr.wroc.pl`; prośba powinna zawierać informację o nazwach i numerach IP co najmniej dwu serwerów obsługujących tę domenę (z zasady jeden z nich powinien leżeć **poza** rejestrowaną domeną, czy precyzyjnie mówiąc powinien znajdować się w innej podsieci).
- Następnie administrator w bazie danych swojej domeny dopisuje rekordy zawierające informacje o nameserwerach obsługujących tę domenę:

  ```
  ...
  immt    IN    NS    ldhpux.immt.pwr.wroc.pl.
          IN    NS    wask.wask.wroc.pl.
  ...
  ```

  Bardzo często oprócz danych o nazwie serwerów pojawiają się informacje o ich numerach IP. Informacja taka jest potrzebna **tylko** wtedy, gdy serwer „mieszka” w delegowanej domenie, a serwer „delegujący” nie jest serwerm wtórnym delegowanej domeny. W przeciwnym przypadku informacje takie nie są potrzebne.
- Trzecim, ostatnim krokiem, jest uruchomienie serwerów w delegowanej domenie.

12.5 Zmiana nazwy domeny

Może się zdarzyć taka sytuacja, że nazwa dotychczasowej domeny przestaje nam odpowiadać (lub musimy ją zmienić ze względów administracyjnych). Właściwie nie jest to wielki problem, ale najczęściej chodzi o to aby rownocześnie przez jakiś czas istniały obie domeny.

Załóżmy, że dokonujemy zmiany z `aaa.pwr.wroc.pl` na `bbb.pwr.wroc.pl`.

- Zakładamy, że serwerem pierwotnym dla obu domen będzie ten sam komputer.
- Zakładamy również, że nie zmieniają się nazwy komputerów w domenie.
- Wszystkie informacje o domenie zapisane będą w jednym pliku o nazwie `db.domena`; wszystkie informacje w tym pliku zapisane są w postaci względnej (jeżeli dotyczą serwowanych domen) i w postaci bezwzględnej (FQDN) w przeciwnym razie.
- Rekord SOA używa znaku „@” dla oznaczenia nazwy domeny.
- W pliku `/etc/named.boot` powinny znaleźć się dwa wiersze:

```
primary          aaa.pwr.wroc.pl          db.domena
primary          bbb.pwr.wroc.pl          db.domena
```

- Jeżeli chodzi o domenę odwrotną, to powinny być wpisane tam odsyłacze do obu nazw lub tylko do tej nowej.

12.6 Zmiana funkcji serwera z *secondary* na *primary*

Opiszemy tu czynności, które należy podjąć w przypadku gdy podejmujemy decyzję o zmianie funkcji serwera z wtórnego (*secondary*) na pierwotny (*primary*). Do takiej sytuacji może dojść gdy „dorośliśmy” już do zajmowania się samemu swoją domeną.

Wszystkie czynności należy przeprowadzić w porozumieniu z administratorem serwera, który dotychczas pełnił funkcję serwera pierwotnego.

Oto schemat czynności, które powinniśmy wykonać:

1. Modyfikujemy plik `named.boot` na komputerze będącym serwerem wtórnym; linię:

   ```
   secondary        domain  156.17.5.2       db.domain
   ```

   zamieniamy na:

   ```
   primary          domain                   db.domain
   ```

2. Modyfikujemy plik `db.domain` (do tej pory zawierał **kopię** informacji z serwera pierwotnego); były w nim linie podobne do:

   ```
   $ORIGIN pwr.wroc.pl.
   domain           IN      SOA     sun2.ci.pwr.wroc.pl. ....
   ```

   (dyrektywa `$ORIGIN` mówi programowi **named** jakiej domeny dotyczą wszystkie dalsze dane). Zastępujemy je linią:

   ```
   @                IN      SOA     myhost.domain....
   ```

   `myhost...` to pełna nazwa komputera serwującego od tej pory informacje o domenie `domain`. Pozostałe informacje powinny być umieszczone zgodnie z zasadami obowiązującymi dla rekordu SOA (patrz str. 12.2.1)

3. W tym samym czasie administrator komputera, który do tej pory serwował informacje dokonuje odpowiednich informacji w pliku `/etc/named.boot` swojego komputera (linię `secondary` zastępuje linią `primary`).

4. Należy zrestartować programy **named** na obu komputerach.

   Uwaga: Praktyka uczy, że restart powinien polegać na „zabiciu” i ponownym uruchomieniu programu **named**. Dokładniej w wersji 4.8.3 każda zmiana w `named.boot` wymaga zabicia i ponownego uruchomienia programu. W wersji 4.9.3 tak radykalna procedura jest niezbędna jedynie w przypadku zamiany serwera głownego na wtórny (i vice–versa) – to jest właśnie nasz przypadek i w przypadku skasowania domeny, gdy serwer pozostaje serwerem autorytatywnym dla domeny wyższej. W pozostałych przypadlach zmiany w `named.boot` mogą być sygnalizowane za pomocą instrukcji:

   ```
   kill -HUP `cat /etc/named.pid`
   ```

12.7 Jak skonfigurować DNS w sieci nie podłączonej do Internetu

Tych informacji w najmniejszym stopniu nie zweryfikowalem. Moja sieć od dłuższego czasu jest podłączona do Internetu i nie będę tego w najbliższym czasie zmieniał `:-)`. Wszystkie informacje zaczerpnęłem z odpowiedniego FAQ[9] [15].

Podstawową sprawą jest skonfigurowanie serwera głównego dla domeny „.” (*root*) i utworzenie odpowiednich rekordów delegujących domenę która nas interesuje (i domenę odwrotną) na ten komputer, który ma nimi zarządzać (wszystkie informacje mogą być oczywiście na jednym komputzerze). W momencie podłączenia - likwidujemy wszytskie „bzdurne” informacje.

Tak więc jeżeli używamy domeny `firma.com.pl` nasz plik `named.boot` będzie wyglał jakoś tak:

```
primary         .               root
primary         pl              pl
primary         com.pl          com.pl
primary         firma.com.pl    firma.com.pl
primary         in-addr.arpa    in-addr.arpa
```

W pliku *root* będzie taka informacja (i oczywiście rekord SOA):

```
pl.     IN      NS      host.firma.com.pl.
```

w pliku pl:

```
com.pl. IN      NS      host.firma.com.pl.
```

i tak dalej (plik firma.com.pl zawiera już ”prawdziwe” informacje - i nie będzie musiał być zmieniany w momencie uzyskania pełnego połączenia – chyba, że zmienią się numery IP).

[9] ftp://ftp.njit.edu/pub/dns/Comp.protocols.tcp-ip.domains.FAQ

Rozdział 13

SNMP na SCO

13.1 Wstęp

Oto drugi rozdział „Podręcznika Administratora”. Powstał on dzięki zaangażowaniu Krzysztofa Murkowskiego[1]. Prawdę mówiąc mój udział sprowadził się jedynie do kionwersji niezbyt długiego listu, który od niego jakiś czas temu otrzymałem.

Poniższe notatki są zawierają jedynie najbardziej elementarne wiadomości na temat konfigurowania SNMP i wykorzystywania go do zdobywania informacji o pracy innego komputera. SNMP wykorzystywane jest przez ogromne pakiety używane do zdalnego zarządzania komputerami w sieci (HP OpenView czy SUN NetNode Manager i wiele innych) ale również do zbierania i prezentowania ruchu sieciowego na interfejsach routerów NASK czy sieci miejskiej WASK.

Wszystkim zainteresowanym poecam również lekturę grupy newsowej comp.protocols.snmp i FAQ tej grupy[2].

13.2 `snmp` – skrypt uruchamiajacy lub zatrzymujacy SNMP

Normalnie powinien odpalać się z `/etc/rc2.d/`. Po wykonaniu powinien wisieć demon - proces `snmpd`.

opcje:

start uruchamia

stop zatrzymuje

Lokalizacja: `/etc/snmp`

13.3 Konfigurowanie serwera, by odpowiadał na niektóre publiczne zapytania

Dane typu osoba kontaktowa, fizyczna lokalizacja serwera wpisuje się do pliku `/etc/snmpd.conf`. Potem należy zatrzymać i od nowa uruchomić `snmpd`. Dane z którego komputera co można pisać albo czytać i z jakim hasłem znajdują się w pliku `/etc/snmpd.comm`.

13.4 Zadawanie zapytań

13.4.1 `snmpstat [option] [host]`

-a RARP table

-i karty sieciowe

-r tablica routingu

-t aktywne połączenia (to samo co ’netstat’)

np: `snmpstat -r 150.254.134.4`

Bez opcji `snmp` wyświetla to samo co z opcja `-t` ale w przyjemniejszej formie.

[1] http://www.ml.put.poznan.pl/~murek

[2] ../toolex/msg0000

13.4.2 `getid` – pobiera dane z SNMP-Agenta

Sciaga 3 zmienne (`sysUpTime.0`, `sysObjectId.0` i `sysDescr.0`)
`getid 150.254.5.248 public`

13.4.3 `getmany` – sciaga z SNMP Agenta wartosci pewnej klasy zmiennych

`getmany 150.254.5.248 public iso` – Wszystkie dane

`getmany 150.254.5.248 public ipRouteDest` – dane `ipRouteDest.`*xxx*

13.4.4 `getroute` – wyswietlenie zdalnej tablicy routingo przez SNMP

Można sprawdzić czyjąś tablicę routingu, np:
`getroute 150.254.5.248 public`
lub `snmpstat -r 150.254.5.248` lub `netstat -r`

Dodatek A

Podstawowe pojęcia związane z transmisją szeregową

Poniższy rozdział został napisany częściowo na podstawie własnych doświadczeń i przemyśleń, częściowo zaś na podstawie znakomiej dokumentacji programu kermit autorstwa Franka da Cruz i Christine M. Gianone [3].

A.1 Format znaków, parzystość

Dane komputerowe przesyłane są zazwyczaj jako ciągi znaków (bajtów) o długości 8 bitów.

Bardzo często, dla sprawdzenia poprawności przesyłanych (przechowywanych) danych uzupełnia się informację o tak zwany bit parzystości. Informacja przechowywana na n bitach uzupełniana jest o $n+1$ bit, którego wartość wyznaczana jest w ten sposób, abyaby sumaryczna liczba jedynek w ciągu $n+1$ bitów była:

- parzysta (ang. *even*),
- nieparzysta (ang. *odd*).

Zatem znak „wielkie A" `1 0 0 0 0 0 1` kodowany jako „zwykłe" ASCII[1] zapisany może być na siedmiu bitach. Ósmy bit w takiej sytuacji może spełniać rolę bitu parzystości, i w konwencji:

parzystej znak zapiszemy jako **0** `1 0 0 0 0 0 1`

nieparzystej znak zapiszemy jako **1** `1 0 0 0 0 0 1`

Czasami stosowano również „pseudoparzystość": $n+1$ bit miał zawsze wartość `0` (ang. *space parity*) albo stale wartość `1` (ang. *mark parity*) i spełniał jedynie rolę „wypełniacza". Bardzo często można bit parzystości wyłączyć i dysponować wszystkimi ośmioma bitami dla potrzeb transmisji danych (ang. *pariy none*).

W czasach nowoczesnych modemów (z korekcją) rola bitu parzystości bardzo spada. Sam modem stosuje pewniejszy protokół nadzoru poprawności przesyłanych danych.

Większość starych wersji systemu UNIX nie była przystosowana ani do pracy z rozszerzonym zestawem znaków ASCII ani do obsługi transmisji danych w trybie bez parzystości. Dziś się to już zmienia.

A.2 Szeregowa transmisja asynchroniczna

Jednym z najbardziej popularnych standardów przesyłania danych cyfrowych na niewielkie odległości jest przesyłanie ich w sposób szeregowy.[2] Standard ten w USA nosi nazwę EIA RS 232 (albo RS 232 w skrócie) a ITV-T V.24 w Europie.

Transmisja nazywana jest **asynchroniczną** gdyż zakłada się, że dane mogą pojawiać się w dowolnej chwili i będą natychmiast transmitowane do odbiorcy.

Z tego też powodu rozpoczęcie transmisji danych musi być zasygnalizowane w jakiś sposób. Używa się w tym celu dwu dodatkowych bitów:

- bit startu – `0`,

[1] „Zwykłe", w odróżnieniu od „rozszerzonego" ASCI obejmuje znaki o kodach od 0 do 127 włącznie.

[2] To znaczy, że wysyłany jest „bit po bicie" w odróżnieniu od transmisji równoległej, gdzie wysyłanych jest kilka bitów równocześnie (wykorzystując w tym celu kilka linii przesyłowych).

- bit stopu – 1.

Zatem znak A wysyłany w sposób szerwgowy „wyglądać” będzie tak:

```
1      _         __
0 __|  |_____|   |
  SP 1 00000 1T
```

Na powyższym schemacie użyto oznaczeń:

S bit startu,

P bit parzystości,

T bit stopu.

A.3 Szybkość transmisji

Nawet gdy transmisja jest asynchroniczna urządzenia nadawcze i odbiorcze muszą być w jakiś sposob „zsynchronizowane” – muszą pracować z „jednakową szybkością” (to znaczy wiedzieć ile „trwa” transmisja pojedynczego bitu).

Szybkośc transmisji podawana jest w bitach na sekundę – **bps** (ang. *bits per second*), a niektórzy (ale nie puryści `:-)` mówią o „bodach”.

Najczęściej używane szybkości w transmisji szeregowej mieszczą sie w następującym szeregu: 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps.[3]

Urządzenia nadawcze i odbiorcze muszą zatem pracować z jednakową szybkością. Zazwyczaj jest ona taka sama w obu kierunkach, choć przed laty używano standardów asymetrycznych.[4]

A.4 „Plex”

Osobną sprawą jest sposób przesyłania danych. Może być on jednokierunkowy (ang. *simplex* albo *one–way*) – to znaczy jedna strona „mówi” a druga „słucha” lub dwukierunkowy (ang. *duplex* albo *two–way*) – strony mogą mówić równocześnie.

Połączenie dwustronne czasami może być realizowane w sposób zwany *half–duplex* albo *one–way alternative*: w danej chwili może mówić tylko jedna strona, do przełączania używany jest specjalny sygnał.

A.5 Sterowanie przepływem danych

W przypadku połączenia dwu urządzeń, które komunikuja się ze sobą może okazać się, że odbiorca nie jest w stanie odebranych danych przetworzyć w temie w jakim one napływają (na przykład drukować ich lub czytać `:-)`.

Musi zatem istnieć metoda pozwalająca na wstrzymanie wysyłania danych. Nazywane jest to **sterowaniem przepływem danych** (ang. *flow control*).

W przypadku transmisji szeregowej bardzo często używany jest sposób, w którym wysłanie znaku `CTRL-S` (ASCII DC3) przez odbiorcę do nadawcy powoduje wstrzymanie transmisji, a znaku `CTRL-Q` (ASCII DC1) jej wznowienie. System taki nazywany jest „programowym sterowaniem przepływem danych” (ang. *software flow control*).

Znacznie wygodniejszy jest „sprzętowy” (ang. *hardware flow control*) sposób sterowania przepływem danych. Wymaga on jednak istnienia dodatkowych linii sygnałowych pomiędzy nadawcą a odbiorcą informacji.

W każdym przypadku wymagane jest aby nadawca i odbiorca wykorzystywał ten sam sposób sterowania przepływem danych.

[3] Zwracam uwagę, że standard RS 232 nie przewiduje szybkości większych niż 20000bps.

[4] Na przykład 1200/75 bps; projektodawcy wyszli z założenia, że powolnemu człowiekowi korzystającemu z klawiatury nie trzeba więcej niż 75 bps (ok. 7 znaków na sekundę), a szybszy komputer wysyłał będzie dane z szybkościa 1200 bps.

A.6 Modem

Maksymalan dozwolona standardem RS232 odległość połączenia wynosi 15 m. Przesyłanie danych na większe odległości wymaga stosowania specjalnych narzędzi.

Przyjęło się (w dokumentacji RS232) urządzenie wysyłające informacje nazywać DTE (ang. *Data Terminal Equipement*), a urządzenie służce do komunikacji DCE (ang. *Data Communication Equipement*).

Modem jest jednym z urządzeń DCE, które zamienia sygnał z postaci cyfrowej do analogowej (o ograniczonym paśmie) nadającej się do przesyłania na większe odległości z wykorzystaniem linii telefonicznycch, na przykład.

Jeżeli używa się „standardowych” (to znaczy takich, do których podłączony bywa zwykły telefon) linii telefonicznych modem może wykonać wszystkie czynności związane z „podniesieniem słuchawki” i wybraniem numeru. Może również odbierać połączenia nadchodzące (po „usłyszeni” sygnału dzwonienia).

Modem nawiązujący połączenie (ang. *originate*) wysyła ton o specjalnej częstości. Po rozpoznaniu go, modem odbierający (ang *answer*) wysyła ton o innej częstości. Sygnały te są nośnikami informacji (ang. *carrier*) i wysyłane są przez cały czas kiedy nawiązane jest połączenie. Przesyłane dane cyfrowe służą do zmiany parametrów czyli modulacji częstości, amplitudy lub fazy sygnałów nośnych.

A.7 Komunikacja komputera z modemem

Komputer korzystający z modemu musi mieć możliwość nadzorowania jego pracy i sterowania nim oraz przesyłania danych. Używanych do tego celu jest 10 linii połączeniowych.

linia	V.24	Nazwa	Kierunek	DB25	DB9	Opis
FG		Frame Ground		1	–	Masa elektryczna
TD	103	Transmit Data	do DCE	2	3	dane z komputera
RD	104	Receive Data	do DTE	3	2	dane z modemu do komputera
RTS	105	Ready To Send	do DCE	4	7	sprzętowe sterowanie przepływem informacji
CTS	106	Clear To Send	do DTE	6	6	sprzętowe sterowanie przepływem informacji
DSR	107	Data Set Ready	do DTE	5	8	DCE pracuje i jest w trybie danych
SG	102	Signal Ground		7	5	masa sygnałowa
CD	109	Carrier Detect	do DTE	8	1	wykrycie nośnej – modemy nawiązały połączenie
DTR	108	Data Terminal Ready	do DCE	20	4	DTE włączony i jest w rybie danych
RI	125	Ring Indicator	do DTE	22	9	Sygnał dzwonienia

Sposób nawiązywania połączenia komputera A z komputerem B (z wykorzystaniem modemu) przebiega w następujący sposób:

1. Komputer A sprawdza czy linia DSR jest aktywna (to znaczy, czy modem jest włączony i gotowy do pracy). Jeżeli nie – błąd, nie można kontynuować połączenia.
2. Komputer A ustawia linię DTS sygnalizując modemowi, że chce rozpocząć komunikację.
3. Komputer wydaje polecenie wybrania numeru.
4. Modem A „podnosi słuchawkę” i sprawdza czy jest sygnał centrali; jeżeli jest – wybiera numer.
5. W przypadku gd nie ma odpowiedzi w zadanym czasie – modem sygnalizuje to komputerowi jako błąd, procedura musi rozpocząć się od nowa.
6. Jeżeli modem B odpowie – rozpoczyna się procedura nawiązania połączenia i ustalenia parametrów transmisji. Jeżeli nie powiedzie się ona w zadanym czasie – błąd.
7. Gdy modemy ustalą szczegóły protokołu przesyłania informacji – A ustawia linię CD informując komputer o nawiązaniu połączenia. Rozpoczyna się przesyłanie danych.
8. Jeżeli komputer wyłączy linię DTR jest to sygnałem dla modemu do przerwania połączenia („odłożenia słuchawki”). Gdy modem A przestanie słyszeć nośną z modemu B – wyłącza linię CD informując komputer A o przerwaniu połączenia.

(Oczywiście zachowanie modemu B i komputera B jest bardzo podobne.)

Bardzo podobne mechanizmy (wykrwanie sygnału CD czy DTR) pozwalają przerwać połączenie telefoniczne w przypadku awarii jednego z komponentów systemu.

A.8 Polecenia rozpoznawane przez modem

Aby komputer mógł się komunikować z modemem, musi istnieć pewien wspólny język. Istnieje nawet pewien międzynarodowy standard, ale mało który modem stosuje się do niego.

Powszechnie używanym standardem jest tak zwany standard poleceń „AT Hayes" (gdyż wprowadził go już dosyć dawno temu znany niegdyś producent modemów Hayes) ale zwracamy uwagę, że implementacje poszczególnych modeli różnych firm potrafia się bardzo różnić między sobą.[5] Zwykle, mniejszy lub większy, podzbiór poleceń obsługiwanych przez modem zawarty jest w jego dokumentacji (i z tego powodu kupując modem, zwłaszcza używany, należy domagać się tej dokumentacji), ale coraz częściej tak zwane modemy „plug and play" nie posiadają praktycznie żadnej dokumentacji, a jedynie drajwer do systemu Windows 3.x/Windows95. Tych radzę się wystrzegać, gdyż mogą sprawiać kłopoty (zwłaszcza używane w innym systemie operacyjnym niż ten wymieniony na pudełku).

Z wymienionych wyżej powodów bardzo trudno wypisać jakikolwiek użyteczny zestaw poleceń wspólnych dla dużej grupy modemów.

Poniżej szereg ogólnych zasad.

Wszystkie polecenia zestawu „Hayes AT" rozpoczynają sie od sekwencji znaków `AT` (od angielskiego słowa *attention* – uwaga). Są one wykorzystywane przez procesor modemu między innymi do określenia szybkości, z jaką komunikuje się z nim komputer (czy ogólniej podłączone do niego urządzenie).

Każde polecenie musi być zakończone naciśnięciem klawisza `<CR>` (Enter, Return). Można grupować kompletne polecenia po kilka w jednym wierszu, ale niektóre, tańsze modemy mają bardzo niewielki bufor w którym zapisują polecenie przed analizą.

Po każdym poleceniu (lub ich ciągu) modem udziela odpowiedzi infomującej o tym czy jego wykonanie zakończyło się poprawnie. Odpowiedź ma postać `OK` w przypadku poprawnego wykonania albo `ERROR` w przypadku błędu. Czasami zamiast odpowiedzi słownych modem udziela ich w postaci liczbowej (`0` dla `OK` lub `1` w przypadku błędu). Czasami odpowiedź jest nieco bardziej komunikatywna: `CONNECT`, `NO DIALTONE`, `BUSY`, `NO CARIER`.[6]

Kilka najbardziej popularnych poleceń zawiera poniższa tabelka.

[5] Doświadczenie uczy, że nie należy zwłaszcza wierzyć reklamowym nadrukom „Hayes compatible" – zazwyczaj nie jest to wcale prawdą!

[6] Odpowiedzi te pojawiają się w procesie wybierania numeru i nawiązywania połączenia z innym modemem; `CONNECT` oznacza, że połączenie nawiązano, `NO DIALTONE` informuje o braku sygnału centrali przed wybraniem numeru, `BUSY` – numer jest zajęty, `NO CARIER` nie udało się uzgodnić szczegółów połączenia lub telefonu nie „odebrał" drugi modem.

Polecenie	Funkcja
`AT`	Nie wykonuje żadnej akcji. Modem odpowiada `OK` jeżeli znajduje się w trybie wykonaywania poleceń.
`AT&C1`	Linia CD jest włączona tylko wtedy, gdy modemy połączą się. Zalecane. `AT&C0` – linia CD włączona zawsze.
`AT&D2`	Modem przerywa połączenie i przechodzi do stanu wykonywania poleceń gdy komputer wyłączy linię DTR. Zalecane. `AT&D0` powoduje ignorowanie przez modem stanu linii DTR.
`ATE1`	Włącza echo. Modem „powtarza każdy znak wysłąny przez komputer. `ATE0` wyłącza tę funkcję.
`ATM0`	Wyłącza głośnik modemu. `ATM1` włącza głośnik modemu tylko na czas nawiązywania połączenia.
`ATQ0V1`	Modem odpowiada używając odpowiedzi tekstowych (`CONNECT`, `OK`) a nie numerycznych.
`ATX0`	Modem odpowiada używając komunikatów z zestawu `OK`, `CONNECT`, `RING`, `NO CARRIER`.
`ATX1`	Modem odpowiada używając komunikatów z zestawu `OK`, `CONNECT`, `RING`, `NO CARRIER`, `ERROR`, `NO DIALTONE`, `BUSY`, `CONNECT 1200`, `CONNECT 2400`.
`ATDT`*nnnnnnnn*	Wybiera numer *nnnnnnnn* używając wybieranie tonowe. Numer może zawierać cyfry oraz odstępy, nawiasy i myślniki – te znaki są ignorowane. Przecinek powoduje wstrzymanie wybierania następnej cyfry na 2 sek. wartość tę można zmienić zmieniając zawartość rejestru `S8`. Litera `W` nakazuje przed wybraniem następnej cyfry odczekać (do 30 sekund) na sygnał centrali. Wykrzyknik oznacza chwilowe (0.5 sek.) zawieszenie (*hook flash*) i następnie powrót do połączenia.[7]
`ATDT`*nnnnnnnn*	Podobnie jak `ATDT`, ale numer wybierany jest sposobem „impulsowym”.
`ATD`*nnnnnnnn*	Numer wybierany jest z użyciem domyślnego sposobu wbierania.
`ATH0`	Powoduje „odłożenie słuchawki” (rozłącza połączenie).
`+++`	Powoduje przejście do trybu poleceń bez przerywania połączenia z innym modemem. Zazwyczaj polecenie jest ignorowane przez modem, jeżeli nie jest poprzedzone dłuższą chwilą ciszy i po jego wydaniu nastąpi również dłuższa chwila ciszy.[8]
`ATO`	Wraca do stanu połączenia ze stanu wydawania poleceń.
`ATZ`	„Zerowane” modemu (to znaczy powrót do pierwonej konfiguracji).
`ATS0=1`	Wprowadza modem w stan odbierania połączeń. 1 oznacza w tym przypadku, że po pierwszym dzwonku modem podniesie słuchawkę. Wpisanie tam $n > 0$ w efekcie spowoduje podnoszenie słuchawki po n–ym dzwonku. Gdy $n = 0$ modem nie odbiera przychodzących połączeń.
`ATS7=`*nnn*	Oczekuje *nnn* sekund na rozpoznanie nośnej.
`ATS8=`*nnn*	Czas przerwy spowodowanej przez przecinek w wybieranym numerze. Standardowo 2 sekundy.
`ATS10=`*nnn*	Opóźnienie pomiędzy zanikiem nośnej a zerwaniem połączenia, podawane w dziesiątych częściach sekundy.
`ATS25=`*nnn*	Czas przez jaki powinna po zmianie utrzymywać się stała wartość linii DTR, aby modem „zauważył” zmianę, podawane w dziesiątych częściach sekundy.
`ATI`*n*	Może wyświetlić różne ciekawe informacje na temat modemu i konfiguracji. Należy próbować różne wartości dla n: 1, 2, ...

A.9 Modulacja, korekcja błędów, kompresja

Modemy komunikują się ze sobą wykorzystując ciągle rosnący zestaw technik modulacyjnych pozwalających na przesyłanie coraz większej ilości informacji przy niezmienionym paśmie przepustowym.[9]

Aby dwa modemy mogły komunikować się ze sobą muszą wykorzystywać conajmniej jedną wspólną technikę modulacji. Załorzyć trzeba również istnienie jakiejś techniki negocjacji wspólnego protokołu.

A.9.1 Sposoby modulacji

Poniższa tabela przedstawia podstawowe sposoby modulacji.

[9]Kanał telefoniczny oferuje pasmo o szerokości nieco ponad 3000 Hz.

Oznaczenie	Rok	Opis
V.34	1994	28800, 26400, 24000, 21600, 19200, 16800, 14400, full duplex
V.32 bis	1991	14400, 12000, 9600, 7200, full duplex
V.32	1984–88	9600, 4800, 2400, full duplex
V.29	1976–84	9600 (dla linii dzierżawionych, MNP6)
V.26 ter	1984–88	2400 full duplex
V.26 bis	1972–84	2400, 1200 half duplex (z 75 bps)
V.22 bis	1984–88	2400 full duplex
V.22	1980–84	1200 full duplex
Bell 212A	1997	1200 half duplex
VA34000	1973	1200 half duplex
V.23	1972–84	600, 1200 half duplex (z 75 bps)
V.21	1964–84	(Europa) 300 full duplex (niezgodny z Bell 103)
Bell 103	?	(USA) 110, 150, 300 full duplex (niezgodny z V.21)

Zazwyczaj procedura połączenia polega na tym, że modem rozpoczyna od najbardziej zaawansowanego sposobu modulacji, a gdy to się nie powiedzie – proponuje prostsze (ang. *fallback procedure*).

A.9.2 Buforowanie szybkości

Mówiąc o szybkości połączenia zwrócić trzeba uwagę na konieczność rozróżnienia dwu pojęć:

- szybkość „modulacji” (wynegocjowana szybkość transmisji danych pomiędzy modemami),
- szybkość interfejsu (szybkość transmisji danych pomiędzy modemem a komputerem).

W najprostszych modemach obie szybkości muszą być identyczne. W takiej sytuacji, jeżeli inicjujemy połączenie z szybkościa 2400 bps, a modemy wynegocjują inna (1200 bps, na przykład) oprogramowanie musi zmienić szybkość przesyłania danych pomiędzy komputerem a modemem.

Współczesne modemy posiadają własność pozwalającą na „buforowanie” szybkości przesyłania danych. W takim przypadku szybkość transmisji pomiędzy komputerem a modemem może być stała i niezależna od szybkości wynegocjowanej przez modemy. Pozwala to, w szczególności, negocjować niestandardowe (to znaczy nie obsługiwae przez układy interfejsu) szybkości „pośrednie”, na przykład 21600 bps.

Komunikat `CONNECT` może (w zależności od skonfigurowania modemu) informować o uzyskanej szybkości modulacji lub o szybkości używanej do komunikacji z komputerem.

A.9.3 Korekcja błędów

Modemy z korekcją błędów pozwalają na polepszenie warunków przesyłania danch na zaszumionych liniach. Zwracamy uwagę, że modem taki może jedynie korygować błędy powstające na linii telefonicznej, a nie na połączeniu pomiędzy komputerem a modemem. Te ostatnie są jednak, na szczęście, znacznie rzadsze.

Pamiętać trzeba jednak o tym, że możliwość korekcji błędów osiągnięta jest za pomocą specjalnego protokołu komunikacji pomiędzy modemami. Jego skutkiem jest przesyłanie informacji w pewnch „blokach” (pakietach). Każdy blok wyposażony jest w nadmiarowe informacje pozwalające automatcznie skorygować pewne błędy albo tylko zauważyć ich powstanie. Prookół uwzględnia możliwość retransmisji błędnie przesłanych danych. Nieuchronnie związane jest to z zajęciem części pasma na te dodatkowe informacje (około 10%). Bardzo często modemy przesyłają dane pomiędzy sobą w trybie **synchronicznym** co pozwala zrezygnować z bitów startu i stopu niezbędnych w trybie asynchronicznym.

Do korekcji błędów używanych jest szereg technik. Najczęściej używane zebrane zostały w tabeli.

Oznaczenie	Opis
MNP Class 1	korekcja błędów, połączenie asynchroniczne, half duplex
MNP Class 2	korekcja błędów, połączenie synchroniczne, full duplex
MNP Class 3	korekcja błędów, połączenie synchroniczne, full duplex
MNP Class 4	MNP 3 + dynamiczna długość pakietów
MNP Class 5	kompresja danch, używane w połączeniu z MNP 4
MNP Class 6	
MNP Class 7	zaawansowana kompresja danych + MNP 4
MNP Class 9	MNP 7 + modulacja V.32
MNP Class 10	MNP 4 zaadaptowane do potrzeb telefonii komórkowej
PEP	opracowany przez Telebit Packet Ensemble Protocol (kombinacja modulacji, korekcji błędów i kompresji danych)
HST	Opracowany przez US Robotics High Speed Technology
V.42	Międzynarodowy standard korekcji błędów
V.42 bis	Międzynarodowy standard kompresji danych

Podobnie jak w przypadku negocjacji sposobu modulacji (szybkości) potrzebna jest procedura negocjacji rodzaju korekcji błędów.

Na rynku dostępna jest pewna grupa modemów których producenci deklarują, że raelizują one korekcję błędów ale za pomocą osobnego oprogramowania. Modemy te mogą działać nawet poprawnie, ale **tylko i wyłącznie** z dostarczonym przez producenta oprogramowaniem. Zazwyczaj próba użycia innego oprogramowania (systemu operacyjnego) zakończy się niepowodzeniem.

A.9.4 Kompresja danych

Ze względu na techniczne ograniczenia (skończone pasmo przepustowe używanych urządzeń telefonicznych) zaczęto w modemach stosowaćróżne techniki kompresji danych mające zwiększyć efektywną przepustowość używanych kanałów połączeniowych.

Oczywiście dane raz już skompresowane (pliki typu `.zip`, `.jpg`, `.gif` i wiele innych) nie mogą być już dalej kompresowane przez modem. Może to prowadzić w przypadku prymitywnych algorytmów kompresji nawet do zwiększenia czasu transmisji pliku już skompresowanego.

Teoretyczna zdolność kompresji danych przez modem możey wynosić 4:1. Stwarza to określone problemy poodczas przesyłania danych.

Załóżmy, że modemy wynegocjowały między sobą szybkość transmisji 14400 bps. Są one w stanie (z użyciem kompresji i w najlepszych warukach) przetransmitować około 57600 bitów w ciągu sekundy (to znaczy taka powinna być szybkość transmisji danych na interfejscie pomiędzy komputerem a modemem). Zatem jeden komputer powinien dane w takim tempie dostarczać, a drugi odbierać – w przeciwnym razie zniweczymy wszystkie efekty kompresji danych.

Szczególną rolę odgrywa w przypadku nowoczesnych modemów (z korekcją i kompresją) synchronizacja przesyłania danych. Niejest rzadkością sytuacja, gdy modemy w pierwszej chwili negocjują szybkość przesyłania 28800 bps, użytkownik chcąć wykorzystać wszystkie zalety kompresji ustawił szybkość interfejsu na 4 x 28800 bps (115200 bps) ale z powodu chwilowego pogorszenia się parametrów połaczenia modemy zmniejszyły szybkośc do 9600 bps i przesyłany jest plik skompresowany. Chwilami modem zajęty jest retransmisją danych. Teoretyczna szybkość przesyłania danych waha się zatem od kilku bajtów na sekundę do ponad 100000 bps.

Najprościej osiągnięć to stosując sprzętową synchronizację przesyłania danych. Osiągamy to stosując:

- odpowiedni kabel połączeniowy z komputerem,
- odpowiednią (praktycznie jak najwyższą) szybkość interfejsu komputer – modem,

po obu stronach połączenia.

A.10 Kable i wtyczki

Odpowiedni koabel powinien zapewniać **pełne** połączenie między modemem a komputerem. Mimo, że możliwe jest połączenie modemu (jak i każdego innego urządzenia korzystającego z portu zeregowego) z komputerem za pomocą kabla zapewniającego połączenie linii TD, RD i SG – nie warto z tego korzystać.

Typowe połączenie DTE i DCE porzedstawia poniższy schemat:

```
             1 FG  *------------------------------* FG   1
             2 TD  *----------------------------->* TD   2
             3 RD  *<-----------------------------* RD   3
  DTE        4 RTS *------------------------------* RTS  4    DCE
(komputer)   5 CTS *------------------------------* CTS  5  (modem)
             6 DSR *------------------------------* DSR  6
             7 SG  *------------------------------* SG   7
             8 CD  *<-----------------------------* CD   8
            20 DTR *----------------------------->* DTR 20
            22 RI  *<-----------------------------* RI  22
```

Jeżeli kompuery znajdują się bardzo blisko siebie – można użyć tak zwanego *null modem cable*, czyli połączenia kablowego zastępującego modem. Odpowiednie połączenia są wówczas następujące:

```
             1 FG  *------------------------------* FG   1
             2 TD  *----------------------------->* RD   3
             3 RD  *<-----------------------------* RD   2
  DTE        4 RTS *------------------------------* CTS  5    DCE
(komputer)   5 CTS *------------------------------* RTS  4  (modem)
             7 SG  *------------------------------* SG   7
             6 DSR *--|
             8 CD  *<-+---------------------------* DTR 20
                                               |--* DSR  6
            20 DTR *---------------------------+->* CD   8
            22 RI  *<-----------------------------* RI  22
```

A.11 Modemy zewnętrzne i wewnętrzne

Kolejny problem to decyzja jaki modem wybrać: zewnętrzny (osobne „pudełeczko” z własnym zasilaczem) czy wewnętrzny (w postaci karty wkładanej do komputera.).

Mimo, iż modem zewnętrzny jest droższy – zalecam właśnie taki. Przemawia za tym kilka powodów:

- modem może być przeniesiony do zupełnie innego komputera (o zupełnie innej architekturze),
- stwarza znacznie mniejsze problemy z konfiguracją,
- posiada kilka lampek (lub nawet wyświetlacz) które pozwalają diagnozować pewne problemy z połączeniem.

Jeżeli już jednak decydujemy się na modem wewnętrzny, warto pamiętać o kilku zasadach:

1. Należy skonfigurować go jako urządzenie COM1 lub COM2 (wyłączając w BIOSie jedno ze standardowo dostępnych na płycie).
2. Jeżeli modem musi pracować jako urządzenie COM3 albo COM4 – należy zadbać, żeby przerwanie IRQ z którego korzsta było przeznaczone wyłącznie dla niego.[10]

I ostatnia uwaga natury technicznej. Warto również zadbać o to, żeby używać układu UART typu 16550.[11]

A.12 Uwagi ogólne

- Do połączenia modemu z komputerem najlepiej używać kabla dostarczonego z modemem lub zalecanego przez producenta.
- Do bezpośredniego połącznia dwu kompterów **bez** pośrednictwa modemu najlepiej użyć dwu sandardowych kabli i specjalnej „przejściówki”.

[10] Porty COM1 i COM2 standardowo wykorzystują przerwania IRQ 4 i IRQ 3. Można (jeżeli tylko karta da się tak skonfigurować) „zabrać” przewanie 7 lub 5 (używane zazwyczaj przez port LPT1) – systemy drukujące sporadycznie korzystają z tych przerwań.

[11] Układ UART odpowiedzialny jest za obsługę portów szeregowych. Starsze komputery wyposażane były badzo często w tańsze układy, które **nie** były wyposażone w specjalne układy buforujące (zwane FIFO). Brak akiego układu praktycznie uniemożliwiał obsługę portu szeregowego z szybkością większą od 9600 bps.

- Jeżeli komputer pracuje w środowisku o dużych zakłóceniach elektrycznych – najbezpieczniej jest użyć kabla ekranowanego.
- Należy używać sprzętowego sterowania przepływem informacji wszędzie tam, gdzie jest to możliwe.
- W przypadku używania modemu z kompresją szybkośc interfejsu powinna być od dwu do czterech razy wyższa od szybkości transmisji pomiędzy modemami.

Zwracamy też uwagę, że mimo iż szerokość pasma oferowanego przez połączenie telefoniczne (ok. 3400 Hz) nie zmienia się, coraz wymyślniejsze techniki modulacji pozwalają na przesyłanie danych z coraz wyższą szybkością.

Gdy już że V.34 to wszystko co można osiągnąć pojawił się „standard" V.34+ zapewniający szybkość transmisji 33600 bps a i następne są już w drodze.

Należy spokojnie oczekiwać na dalszy rozwój wydarzeń.

Dodatek B

Podziękowania

Bardzo serdecznie pragnę podziękować wszystkim tym, którzy podzielili się ze mną swoimi uwagami o tekście, oraz tym, którzy zechcieli wskazać na błędy i niedociągnięcia tekstu, a zwłaszcza:

- Tomkowi Surnaczowi[1] (tsurmacz@ict.pwr.wroc.pl)
- Szymonowi Sokolowi[2] (szymon@uci.agh.edu.pl)
- Jurkowi Pankiewiczowi[3] (pankiewicz@pwr.wroc.pl)

W tym podręczniku umieściłem odsyłacze do materiałów dostępnych w sieci, a przygotowanych przez innych jej użytkowników. Pragnę podziękować wszystkim osobom z wyników pracy których korzystałem, a w szczególności:

- Krzysztofowi Krupie, niegdyś administratorowi serwera WWW w Instytucie Materiałoznawstwa i Mechaniki Technicznej Politechniki Wrocławskiej; z jego pomocy korzystałem bardzo często, niestety, wybrał inne zajęcie.
- Ryszardowi Bernardynowi z Centrum Informatycznego Politechniki Wrocławskiej i Izydorowi Statkiewiczowi z Biblioteki Głównej Politechniki Wrocławskiej, autorom dokumentacji programu **pmail**.
- Piotrowi Piątkowskiemu z Uczelnianego Centrum Informatyki Akademii Górniczo Hutniczej w Krakowie.
- Włodzimierzowi Macewiczowi z Poliechniki Warszwskiej.
- Jerzemu Kędrze z Krakowa,
- Piotrowi Gronkowi z AGH w Krakowie,
- Krzysztofowi Murkowskiemu za rozdział dotyczący SNMP w systemie SCO UNIX.

Osobne podziękowania należą się Krzysztofowi Murkowskiemu[4], który zdecydował się podesłać do opublikowania swoje notatki na temat SNMP.

[1] http://www.ict.pwr.wroc.pl/people/tsurmacz.html
[2] http://www.uci.agh.edu.pl/~szymon/
[3] mailto:pankiewicz@pwr.wroc.pl
[4] http://www.ml.put.poznan.pl/~murek

Spis Literatury

[1] N. Borenstein. A user agent configuration mechanism for multimedia mail format information. RFC 1343, June 1992.

[2] N. Borenstein and N. Freed. Mime (multipurpose internet mail extensions): Mechanisms for specifying and describing the format of internet message bodies. RFC 1341, June 1992.

[3] Frank da Cruz and Christine M. Gianone. *Using C–Kermit Communication Software*. Digital Press, Newton, MA, 1997. ISBN 1-55558-164-1.

[4] Daniel P. Dern. Plugging into the Internet. *Byte*, 17(10):149+, October 1992.

[5] *PC/TCP Network Software for DOS. User's Documentation.* FTP Software, Inc., Wakefield, MA, July 1992. Dokumentacja techniczna oprogramowania.

[6] D. Goldsmith and M. Davis. Using unicode with mime. RFC 1641, July 1994.

[7] Craig Hunt. *TCP/IP Network Administration.* An O'Reilly & Associates, Inc., 103A Morris Street, Sebastopol CA 95472, March 1993. Niezla ksiazka opisujaca konfiguracje oprogramowania sieciowego na komputerze UNIXowym.

[8] Anna Trykozko i Micha Jarski. *Poczta elektroniczna – tajemnice Elm.* Zaklad nauczania informatyki MIKOM, Warszawa, 1994. ISBN 83-85625-62-3.

[9] Brendan P. Kehoe. *Zen and the Art of the Internet. A Beginner's Guide to the Internet.* Prentice Hall, Englewood Cliffs, NJ, second edition, 1993. Interesujacy przewodnik po Sieci zawierajacy wiele informacji na temat jej wykorzystania. Dostepny w postaci zbioru typu `.tex` lub `.ps`.

[10] J. Klensin, N. Freed, M. Rose, E. Stefferud, and D. Crocker. Smtp service extensions. RFC 1651, July 1994.

[11] Ed Krol. *The Whole Internet User's Guide & Catalog.* A Nutchell Handbook. O'Reilly & Associates, Inc., Sebastopol, CA, dec 1992. Znakomity przewodnik po Sieci, zawierajacy mnostwo interesujacych informacji.

[12] Wlodzimierz Macewicz. *Jezyk skladu dokumentu hipertekstowego HTML i HTML+.* Zaklad nauczania informatyki MIKOM, Warszawa, wrzesien 1995. Bardzo interesujaca ksiazka na temat elementow jezyka HTML.

[13] *Net forum, dodatek do PC Kuriera*, 3(25), grudzien 1992. Informacje na temat sieci lokalnych i globalnych.

[14] *Net forum, dodatek do PC Kuriera*, 4(3), luty 1993. Informacje na temat sieci lokalnych i globalnych. Struktura sieci w Polsce, Internet w Polsce.

[15] Chris Peckham. internet/tcp-ip/domains-faq. Znakomite zrodlo informacji na temat zarzadzania DNS, April 1994.

[16] Jerry Peek, Tim O'Reilly, and Mike Loikides. UNIX *Power Tools.* An O'Reilly & Associates/Bantam Book, 103A Morris Street, Sebastopol CA 95472, March 1993. Znakomita ksiazka na temat UNIXA.

[17] Jonathan B. Postel. Simple mail transfer protocol. RFC 821, August 1982.

[18] John S. Quarterman. Recent internet books. Request for Commnents: 1432, March 1993. nic.ddn.mil:/rfc/rfc1432.txt.

[19] Kevin Reichard. Pan.Listonosz@Internet, pocztowe oprawy komunikacyjne dla Internetu. *PC Magazine po polsku*, (7/29):53+, July 1995. Ciekawe omowienie kilu komercyjnych pakietow poczty elektronicznej, pracujacych glownie w srodowisku MS–Windows.

[20] Paul Vixie. Name server operations guide for BIND release 4.9.3. Rozpowszechniany wraz ze zrodlami **bind**, 1994.

[21] Merik Voswinkel. Locating internet access in Europe. *Byte*, 17(10):151, October 1992.