Jerzy A. Supel

Jak zacząć z UNIXem[1]

17/1994

[1] Dla użytkowników systemu operacyjnego SunOS i SOLARIS

WARSZAWA 1994

ISBN 0208-5658

Praca wpłynęła do Redakcji dnia 8 kwietnia 1994 r.

Na prawach rękopisu

Instytut Podstawowych Problemów Techniki PAN
Nakład 200 egz. Ark. wyd. 4,0 Ark. druk. 5,00
Oddano do drukarni w czerwcu 1994 r.

Wydawnictwo Spółdzielcze sp. z o.o.
Warszawa, ul. Jasna 1

Wstęp

Moje obserwacje, pewnie niezbyt odkrywcze, wykazują, że poczta elektroniczna jest najszerzej wykorzystywaną aplikacją UNIXową. Możliwość korzystania z poczty przyciąga do komputera nawet jego dotychczasowych przeciwników. Dla innych jest bodźcem do pogłębienia powierzchownej lub niepełnej wiedzy o systemie. Poczta jest, jak na razie, "motorem" rozwoju sieci komputerowych, należy więc poświęcić jej odpowiednio dużo uwagi. Dalej mam zamiar w miarę szczegółowo opisać zasady korzystania z programu `mail`, który jest tak powszechnie wykorzystywany.

Właściwie mógłbym ograniczyć się do tego tylko rozdziału, a i tak wielu użytkowników UNIXa byłoby usatysfakcjonowanych. Często słyszę: *ach, przydał by się krótki opis poczty.* Albo coś w tym rodzaju: *brak poręcznego spisu komend.* Otóż, co jak co, ale poczta opisana jest w systemie bardzo wyczerpująco. Wystarczy tylko `man mail` i na ekranie będzie coś ze 40 stron. A ja wciąż słyszę te westchnienia: *przydało by się ...*, czyli to nie wystarcza. Dlatego zabrałem się za pisanie. Niestety, aby korzystać z poczty elektronicznej trzeba przynajmniej umieć zacząć sesję UNIXową, no i parę innych jeszcze rzeczy. Musiałem dodać kilka jescze rozdziałów. I tak się zaczęło.

Zawartość tej książeczki jest głównie kompilacją różnych instrukcji obsługi i opisów funkcji, wzbogacona miejscami o doświadczenia z pracy w sieci SUN IPPT. Wyboru dokonałem na podstawie obserwacji pracy użytkowników tej sieci. Założyłem, że wielu Czytelników nie będzie czytać tego opracowania strona po stronie, stosuję zatem wiele odsyłaczy oraz skoków "w przód i w tył", co może nie jest eleganckie ale ułatwia pisanie.

Tych, którzy chcieliby zapoznać się z UNIXem bardziej metodycznie odsyłam do klasycznej już, na rynku polskim, pozycji *System operacyjny UNIX™*, Peter P.Silvester (WNT Warszawa 1991). Za najlepsze wprowadzenie do pracy uważam *UNIX dla opornych*, J.R.Levine i M.Levine (wyd. IDG Books, 1993r. edycja polska), trudno przecenić tę książkę napisaną bardzo przystępnie, z troską o poprawność a na dokładkę z humorem. Polecam też *Petera Nortona przewodnik po UNIX-ie* autorstwa Petera Nortona i Harley Hahn. Metodyczny i bardzo obszerny, z odniesieniami do doświadczeń z DOSem, choć dla mnie sztucznymi.

Dla bardzo dociekliwych pozostaje zawsze pełna dokumentacja systemu SunOS 4.1.x (12 tomów, ok.10 tys. stron) lub Solaris 2.2 w 53 tomach (ok. 110 cm półki), choć i tam nie wszystko można znaleźć. Na szczęście dostęp do Internetu stwarza nieograniczone możliwości otrzymania informacji o systemach UNIXowych.

Konwencja typografii

Opisując sosób korzystania z poleceń UNIXA starałem się przestrzegać pewnych zasad dla odróżnienia tekstu od komend i komunikatów na ekranie:

- **Pismo wytłuszczone** oznacza to, co wpisujemy z klawiatury w przykładowych sesjach roboczych:

  ```
  lksj% date
  ```

  Dodatkowo owalna ramka przypomina o tym, że tekst poniżej ramki jest tym co widać na ekranie terminala.

- **Pismem bezszeryfowym** wyróżnione są, z tekstu otaczającego, polecenia systemowe:

  > Możesz stać się superużytkownikiem przez wpisanie root po znaku `login:` rozpoczęcia sesji lub przez wpisanie komendy su po znaku zachęty % lub $.

- `Courier` **jest używany do listowania programów, nazw programów lub nazw komputerów. Dla przykładu:**

  > Otoczenie *C shell* ustawiane jest indywidualnie dla każdego użytkownika w zbiorze `.cshrc`

  a także do zaznaczenia tekstów wyświetlanych na monitorze przez system:

  ```
  $ who
  jsupel      console      Mar 11 15:21
  ```

notki na marginesie wskazują bardziej istotne elementy tekstu

- *Pismem pochyłym* wyróżnione są argumenty poleceń, zmienne i tytuły książek. Dla przykładu:

  > Wpisz z klawiatury *nazwa_zbioru* jako argument tak jak opisano w *SunOS 5.2 Reference Manual.*

Gdy zaznaczono użycie kombinacji dwu klawiszy oznacza to, że należy trzymać wciśnięty pierwszy klawisz a następnie wybrać drugi i przycisnąć. Dla przykładu Ctrl-D oznacza, że trzymając wciśnięty klawisz Ctrl przyciskamy klawisz D.

> *Uwaga! Mimo, że w powyższym przykładzie* Ctrl-D *jest użyta duża litera* D *to nie używamy klawisza* Shift. *Ma to być kombinacja dwu klawiszy, oznaczonych na klawiaturze jako* Ctrl *oraz* D.

Rozdział 1

Zaczynamy!

Wiele zależy od urządzenia przy którym zasiadłeś. Może to być stacja robocza (komputer z monitorem i klawiaturą) lub terminal (tylko monitor z klawiaturą). Monitor może być monochromatyczny znakowy (wyświetla tylko cyfry, litery i pewne znaki specjalne) ale też może być kolorowy z możliwością wyświetlania grafiki (obrazków). Komputer jest albo typu work stations (np. Sparc2 lub Sparc SLC) albo PC (AT, 386, 486 a nawet XT). Ponieważ system Solaris jest jedną z implementacji standardu UNIX czasami będę odwoływał sie tylko do tej nazwy, wymiennie z nazwą Solaris. UNIX jest systemem operacyjnym takim jak DOS i inne. Tyle, że o rozbudowanych możliwościach i pozbawiony wielu wad i ograniczeń DOSa. W opinii początkujących użytkowników uchodzi za "trudny" i "mało przyjacielski" w porównaniu z DOSem. Jest to prawdą, ale nie do końca. Zagłębianie się w DOSie też wymaga wielu umiejętności. Na codzień jednak, ci którzy pracują w otoczeniu DOS w rzeczywistości mało go używają, uruchamiając od razu swoją aplikację, która ingeruje w jądro systemu lub przechodzą, wręcz, do innego systemu jak np. MS Windows. Podobnie jest w przypadku UNIXa, też korzystamy z aplikacji lub GUI (Graficznego Interfejsu Użytkownika) jakimi są OpenWindows lub X-Windows. Ale o tym później, teraz chcemy zacząć pracę "w UNIXie". *UNIX a DOS*

Jest kilka możliwości rozpoczęcia sesji UNIXowej zależnie od rodzaju urządzenia[1.3] przy którym zasiedliśmy a także od ustawienia środowiska i zmiennych systemowych.

Różnice na starcie, do chwili zgłoszenia się systemu przez wyświetlenie na ekranie:

```
login:
```

są pochodną różnic sprzętowych. Zależnie od tego czy jest to PC, Work Station, X-terminal bądź inne urządzenie, w różny sposób są one przygotowywane do pracy. Jednak sam proces "logowania" czyli otwierania sesji roboczej jest zawsze jednakowy.

1.1 Otwarcie sesji roboczej

login name
password

Każdy użytkownik UNIXa ma nazwę użytkownika [login name] i hasło [password], są jemu przypisane przez administratora systemu. Jeśli jesteś dopiero kandydatem do pracy z systemem i nie masz jeszcze założonego konta należy zwrócić się z tym do administratora systemu.

Jeśli już znasz nazwę użytkownika [login] i hasło [password], a na ekranie widnieje napis:

```
login:
```

wpisz z klawiatury nazwę użytkownika, np.:

```
login: jsupel
```

i naciśnij *Enter* (czasami ten klawisz jest opisany *Return*), system odpowie żądaniem hasła:

```
passwd:
```

tu należy wpisać znane nam hasło i znów nacisnąć Enter.

> ***Uwaga! Przy wpisywaniu hasła system zachowuje się dyskretnie i nie wysyła echa na ekran.***

błędy hasła

Zdarza się, że popełnimy pomyłkę w nazwie lub haśle (najczęściej zmienimy jedną literkę lub dodamy zbędną spację przed lub po nazwie lub haśle, bywa że zapomnimy hasła itp. (strasznie dużo możliwych błędów przy tych dwu wyrazach), system wtedy odpowie:

```
login incorrect:
```

i trzeba zaczynać zabawę od nowa. Nawet jeśli pomyłka nastąpiła w nazwie, system najpierw pyta o hasło, a dopiero po tym informuje o błędzie. Warto o tym pamiętać i sprawdzić poprawność nazwy użytkownika a nie tylko hasła. Jeśli nasze konto nie jest związane z hasłem (są takie przypadki np. przy otwieraniu sesji z FTP[6.0] serwerem, lub inne dopuszczone przez administratora) to system otwiera się zaraz po podaniu nazwy konta, nie żądając podania hasła.

Jeśli nie popełnimy pomyłki przy wpisywaniu nazwy i hasła to system rozpoczyna sesję wyświetlając kilka linijek informacji, np.:

```
Last login: Tue Apr 12 14:44:15 from lkpc2
Sun Microsystems Inc.  SunOS 5.3 Generic September 1993
You have mail.
lks%
```

Znak zachęty (przyzwolenia) [prompt] – w tym przypadku `lks%` – rozpoczyna ostatni wiersz. Po tym znaku można już wpisywać polecenia systemowe.

Jeśli uruchomiliśmy sesję na terminalu graficznym, może się zdażyć, że system w sposób automatyczny uruchamia interfejs graficzny (np. Open Look lub Motif). Wtedy consolą (dla uruchamiania poleceń systemowych "z klawiatury") staje się jedno z okienek, np. Xterm lub Command Tool.

> *Uwaga! Sposób pracy w otoczeniu okien nie jest szerzej omawiany w tym opracowaniu.*

1.2 Zamknięcie sesji roboczej

Po zakończeniu swojej pracy, jeśli chcesz zamknąć otwartą sesję, wpisz:

```
$ exit
```

jeśli pracujesz w otoczeniu Bourne shell, lub:

```
% logout
```

jeśli pracujesz w otoczeniu C chell (ale **exit** też zadziała).

> ***Uwaga! Próba odłączenia się od systemu przez odłączenie zasilania od komputera powoduje komplikacje trudne do przewidzenia. Z całą pewnością narazisz się na nerwową reakcję administratora systemu, któremu zadasz dodatkowej, niemiłej pracy. Sobie możesz skomplikować sytuację przez stratę swoich zbiorów.***
>
> ***Odłączenie zasilania, w trakcie sesji, od terminala nie jest tak brzemienne w skutki (chyba, że jest to consola serwera z otwartym kontem superużytkownika, ale Ciebie przecież to nie dotyczy).***

Prawidłowo zakończona sesja powoduje zamknięcie wszyskich otwartych zbiorów a na ekranie ukaże się znak zachęty `login:`

Wyłączać zasilanie?

Wygaszanie terminala (wyłączanie zasilania) po zakończeniu sesji nie jest konieczne, wyłączanie zasilania komputera (zwłaszcza typu *Work Station* nie jest zalecane (zwłaszcza jeśli jest to komputer włączony do sieci i nie mamy pewności czy ktoś nie używa go zdalnie).

Opisany wyżej sposób zamknięcia sesji roboczej (wyjścia z systemu) jest realizowany w trybie pracy z terminalem alfanumerycznym. W przypadku pracy w trybie okienkowym zamknięcie konsoli nie jest automatycznie wyjściem z systemu. W trybie okienkowym należy ustawić wskaźnik myszy na wolnym polu ekranu (nie zajętym przez okno), prawym przyciskiem myszy otworzyć menu *Work Space* i wybrać opcję *Exit*.

> *Uwaga! Tryb pracy z okienkami nie jest szerzej omawiany w tym opracowaniu.*

1.3 Różnice w sposobie otwarcia sesji

Jeśli jest to PC,
o którym wiemy, że jest przyłączony do sieci UNIXowej.

"PCet" terminalem

Zdecydowana większość PC po włączeniu zasilania zgłasza się użytkownikowi w sposób właściwy dla DOS czyli na ekranie będzie coś w rodzaju C:\ > (Pomijam tu nieliczne jeszcze przypadki innego systemu zainstalowanego na PC, w tym systemów UNIXowych. Dla porządku dodam, że Solaris 2 istnieje także w wersji PCtowej). Trzeba więc powiedzieć DOSowi, że chcemy przyłączuć sie do UNIXa. Wystarczy uruchomić DOSową aplikację o nazwie **telnet** (lub inny program nazywany emulatorem terminala) i już. Czasami ta aplikacja może się nazywać inaczej (np. **tn**) ale to będzie wiedział ktoś kto opiekuje się tym właśnie komputerem. A więc dla przygotowania komputera PC do rozpoczęcia sesji UNIXowej należy wykonać polecenie DOSowe, np.:

telnet

```
C:\ >tn sc2000
```

Argumentem polecenia **tn** jest nazwa sieciowa maszyny do której chcemy się przyłączyć. W tym przypadku jest to **sc2000** ale może być **lkt** lub **lka1** (są to nazwy innych komputerów w sieci SUN IPPT), można zapytać kogoś lepiej zorientowanego do jakiej maszyny najlepiej się przyłączyć. (Już po rozpoczęciu sesji, używając polecenia **rup**, można dowiedzieć się jakie aktualnie komputery są włączone do sieci). Po wykonaniu polecenia **telnet** nastąpi przyłączenie PC do maszyny UNIXowej o czym system poinformuje nas wyświetlając odpowiedni komunikat na ekranie. Na zakończenie "przyłączania" zostanie wyświetlony znak przyzwolenia rozpoczęcia sesji:

```
login:
```

po którym możemy rozpocząć sesję w sposób właściwy dla systemu UNIX. Należy jeszcze dodać, że od tej chwili nasz PC staje się terminalem systemu UNIX, będzie zachowywał się inaczej niż "pod DOSem". Do czasu kiedy nie uruchomimy innej niż **telnet** aplikacji terminal jest terminalem znakowym, wyświetla komunikaty na ekranie i przyjmuje polecenia z klawiatury. Pewne funkcje, właściwe tylko dla aplikacji **telnet**, można uzyskać przez użycie kombinacji klawiszy. Funkcje te opisane są na ekranie pomocy, który ujawnia się po naciśnięciu *Alt H* (dwa klawisze równocześnie). Ponieważ terminal nie wykonuje programów (tylko wyświetla wyniki programów uruchomionych w innym komputerze), wyłączenie go nie powoduje tych straszliwych problemów jakie mogą powstać przy niewłaściwym wyłączeniu stacji roboczej.

terminal znakowy

Jeśli jest to komputer typu stacja robocza
[Work Station] (nazywany także stacją UNIXową).

Komputery UNIXowe z zasady nie są odłączane od zasilania bez wyraźnej potrzeby. Uruchomienie takiego komputera "od zera", aczkolwiek wydaje się,

że przebiega automatycznie, zawsze może sprawić kłopot początkującemu. Jeśli chcesz zacząć pracę z systemem lepiej aby był on już włączony do zasilania, a na ekranie, w ostatniej linii wyświetlanego tekstu, znajdował się znak przyzwolenia wejścia do systemu. Może też być to jaśniejszy prostokąt po środku ekranu z napisami `login:` oraz `pasword:` *Stacja robocza*

> Uwaga! Ciemny ekran monitora nie musi oznaczać, że komputer jest wyłączony! Dotknij klawiatury (naciśnij spację lub Enter) a komputer prawdopodobnie ożyje. Dla zaoszczędzenia luminoforu lampy kineskopowej jest ona wyłączana samoczynnie po kilku minutach bezczynności klawiatury, komputer zasypia. Budzi się natychmiast po dotknięciu klawiatury. *ciemny ekran*

Jeśli więc osoba, która poprzednio używała tego komputera, odłączyła się od systemu w sposób poprawny, to ostatnią linią tekstu wyświetlanego na ekranie jest:

`login:`

a to daje nam szansę na rozpoczęcie pracy.

1.4 Konto, hasło, sesja

Standardowa sesja jest definiowana jako interwał od momentu przyłączenia się użytkownika do systemu [log in] a chwilą odłączenia [log out]. System jest wielodostępny (wielu użytkowników równocześnie korzysta z tych samych zasobów lub z tego samego procesora), posiada więc wbudowane mechanizmy obrony przed nieupoważnionym użyciem. Za każdym razem gdy otwieramy sesję roboczą sprawdzane są prawa dostępu. Nazwa (konto) użytkownika w systemie [login name, user name, account] służy jako znak rozpoznawczy dla systemu i innych użytkowników systemu. Hasło [password] ogranicza dostęp do konta do osób znających to hasło. Hasła nie zna nikt więcej niż te osoby, którym zostało ono ujawnione. Nie zna go także operator systemu, choć niektórym wydaje się to nieprawdopodobne. Operator systemu może tylko wymazać hasło z systemu ale nie może go podejżeć. Możliwe jest oczywiście rozszyfrowanie hasła, zwłaszcza jeśli jest to ciąg liter znajdujący się w słowniku lub nazwa użytkownika. *Nazwa użytkownika*

1.5 Shell, Prompt

Znak zachęty [prompt] może być dowolnym znakiem (ciągiem znaków) ustalonym w zbiorze zawierającym opis zmiennych systemowych[9.1] [environment variables]. Są jednak pewne zwyczaje i znaki nie są całkiem dowolne, mają pomagać w poruszaniu się po systemie. *Znak zachęty*

Gdy po raz pierwszy przyłączasz się do systemu (albo otwierasz nowe okno Command Tool lub Shell Tool) i widzisz znak zachęty [prompt] oznacza to, że interpreter poleceń [shell program] został uruchomiony automatycznie. Ta

powłoka (tak będę dalej nazywał interpreter poleceń) nazywana jest powłoką otwarcia [login shell] a o jej typie zadecydował administrator systemu w chwili otwierania Twojego konta.

C shell *Bourne shell*

Wyjaśnienia wymaga różnorodność programów shell (powłoki). Znane są conajmniej trzy podstawowe [C shell, Bourne shell, Korn shell], wszystkie reprezentowane w systemie Solaris. Różnią się na tyle, że należy być zorientowanym, który z nich jest aktywny. Domyślnym w systemie Solaris 2 jest shell Bourne'a [Bourne shell] jednak częściej używanym (ze względu na przyzwyczajenia z UNIXa BSD) jest shell C [C shell]. Dla uzewnętrznienia różnicy między powłokami przyjęły się różne znaki zachęty.

% najczęściej oznacza C shell

$ najczęściej oznacza Bourne shell

Powyższa zasada nie jest wymogiem systemu a jedynie umową i jest możliwy inny znak zachęty. W szczególności do znaku zachęty (jak wyżej) może być dodana nazwa maszyny (np `lksj`), której procesor odbiera rozkazy powłoki.

Wtedy znak zachęty może wyglądać następująco:

```
lksj%          (C shell)
```

```
lksj$          (Bourne shell)
```

Ten sposób wyświetlania znaku przyzwolenia jest szczególnie przydatny w sytuacji pracy w sieci, gdy isnieje możliwość rozpoczęcia sesji na różnych komputerach.

Może zdażyć się, że spotkasz sie z wieloma innymi znakami przyzwolenia. Np. możesz ujżeć na ekranie znak "hach" jako znak zachęty:

```
#
```

Pijany Zając

co oznacza, że rozrabiasz w systemie jak pijany zając (stałeś sie superużytkownikiem, mam nadzieję, że świadomie). Lepiej opuścić ten tryb pracy przez wpisanie komendy exit lub *Ctrl-D.*

O innych znakach zachęty (w tym **&** właściwym dlą poczty) trochę później.

Rozdział 2

Jak uruchamiać polecenia systemowe

Jeśli już system rozpoznał w nas uprawnionego użytkownika możemy wydawać mu polecenia wykonania czegoś tam.

Zwróć jednak uwagę na znak zachęty[1.4] i pamiętaj, że nie wszystkie komendy Bourne shell (znak $) da się uruchomić w C shell (znak %) i odwrotnie.

2.1 Wpisanie polecenia

Widzimy już znak przyzwolenia [prompt], to znaczy, że system czeka na nasze rozkazy. Spróbuj wpisać (używając klawiatury) polecenie date tak jak pokazano na poniższym przykładzie i naciśnij klawisz Enter

```
$ date
Mon Mar 7 18:46:51 MET 1994
$
```

Jak widać polecenie spowodowało wyświetlenie aktualnej daty i czasu, nawet z podaniem strefy czasowej - MET [Middle Europian Time] (przy okazji wiadomo kiedy pisałem tę linijkę).

Po wykonaniu zadania system oczekuje na następne polecenie (znak zachęty **$**). Brak znaku zachęty może znaczyć, że system jeszcze nie wykonał zadania. Może ale nie musi, tak już z UNIXem jest. Czasami trzeba cierpliwie poczekać aż system poinformuje o zakończeniu pracy. Zadanie może być proste ale ... jakieś odwołania do dysków sieciowych ... jakieś czekanie w kolejce ... duże obciążenie sieci ... demony! (tak, demony czyli procesy systemowe, żyjące własnym życiem) wszystko to miesza i utrudnia pracę. Czasami więc trzeba poczekać, aż system znów pozwoli się zatrudnić. Oczywiście są na to sposoby aby wykonywać kilka zadań równocześnie, w końcu jest to UNIX, ale o tych sposobach później[2.5].

brak znaku zachęty

Spróbuj tej samej komendy date ale zacznij dużą literą:

```
$ Date
Date:  Command not found
```

duże i małe litery

Z tego widać, że UNIXowi nie jest wszystko jedno: duże czy małe litery. Poza tym jest na tyle uprzejmy, że informuje czego nie zrozumiał. Inną sprawą jest, że my nie zawsze rozumiemy czego on nie rozumie.

Najczęściej przyczyną błędu jest tzw. literówka. Trzeba więc wpisać polecenie jeszcze raz. Jeśli błąd zauważyliśmy przed wydaniem polecenia wykonania (*Enter*) to mamy szansę się poprawić. Znane nam klawisze *Delete* i *Backspase* czasami działają, czasami nie (możesz spróbować). Czasami można nawet przesuwać kursor strzałkami po linii edycji, bez kasowania, ustawić go na złą literkę i ciach! skasować. Ale tak naprawdę to niezawodne jest tylko *Ctrl-U* (no, prawie zawsze).

kasowanie znaków

2.2 Jak wpisać długą komendę

Czasami komendy (polecenia) są długie bo zawierają opcje, parametry i inne bajery. Co zrobić jeśli nie mieszczą się w 80 znakach? Bez paniki proszę. Wstawiamy \ (backslash) i kontynuujemy w nowej linii:

```
$ date; \
 logname
Mon Mar 7 19:16:51 MET 1994
jsupel
$
```

rozdzielić polecenia

Przy okazji sprawdziliśmy, że można podać równocześnie dwa polecenia, rozdzielić je trzeba jednak średnikiem.

2.3 Powtórka polecenia

!! wykrzykniki

Czy trzeba wpisywać od nowa całe polecenie jeśli chcemy je powtórzyć? Niekoniecznie. UNIX trzyma w pamięci wydane mu (i wykonane, te z błędami też) polecenia, a C shell wykorzystuje to. Wystarczy wklepać dwa wykrzykniki !! i przybić *Enter* a znów wykona się poprzednie polecenie (załóżmy, że ostatnim poleceniem było date):

```
lksj% !!
date
Mon Mar 7 19:26:32 MET 1994
lksj%
```

Można przywołać z pamięci i wykonać każde inne wcześniej wykonane polecenie przez wpisanie !*x*, gdzie *x* jest numerem polecenia na liście *historii*. Jak pamiętać numer? Nie trzeba, wystarczy wydać UNIXowi polecenie: *!x*

```
lksj% history
 1 pwd
 2 clear
 3 date
 4 logname
 5 date
 6 history
```

i już znamy numery wcześniejszych poleceń (liczba pamiętanych poleceń jest ustawiana w zbiorze `.cshrc`[9.1]). *history*

A jeśli chcemy powtórzyć tę komendę, która była wykonywana dwie wcześniej (to się czasami pamięta), to wystarczy odjąć dwa, czyli:

```
lksj% !-2
logname
jsupel
lksj%
```

!abc

A jeśli i tego za mało to jest jeszcze inna metoda. Po wykrzykniku napisz kilka pierwszych znaków komendy, którą chcesz powtórzyć np. !cl i *Enter*. Wykona się polecenie clear bo wcześniej było wykonywane. Dobre, prawda. Pamiętaj, że to tylko C shell taki dobry (ten z %).

2.4 Potoki? Strumienie?

Zwykle komputer (system) komunikuje się z Tobą pobierając rozkazy z konsoli (klawiatury) i wysyłając komunikaty na konsolę (ekran). Nie ma tu sprzeczności nazwy, logicznie konsola jest jednym urządzeniem, fizycznie rozdzielona na dwa elementy. Czy zawsze obowiązuje taki przepływ informacji? Oczywiście nie. Możesz np. skierować strumień danych wyjściowych bezpośrednio do zbioru zamiast na ekran. Poniższy przykład ilustruje użycie symbolu > przełączającego urządzenie wyjścia: *przełączanie wyjścia*

```
$ date > zbiorek.z.wynikami
$
```

Dane powędrowały do nowo utworzonego zbioru `zbiorek.z.wynikami` (system sam zadbał aby otworzyć zbiór o tej nazwie) zamiast na ekran. To działa w obie strony! System może wczytywać ze zbioru (zamiast z klawiatury) potrzebne mu dane (ale pamiętaj o odwróceniu znaku <).

Jeśli wcześniej istniał zbiór o tej nazwie to dane w nim zawarte zostaną przykryte nowymi danymi (mówiąc wprost: zostaną skasowane). Jeśli zbiór już istnieje a chcesz dołączyć do niego dane to zamiast symbolu > użyj >>.

dołączanie danych

Przy okazji przekonaliśmy się, że UNIXowi jest wszystko jedno ile kropek jest w nazwie zbioru oraz ile znaków jest przed i za kropką. O zbiorach powiem więcej ale później[3.1].

Jeśli chcesz teraz sprawdzić co jest w zbiorze `zbiorek.z.wynikami` to jest bardzo użyteczne polecenie **more** (bardzo lubię to polecenie, więc później[3.1] napiszę o tym coś więcej), które listuje zawartość zbioru:

```
$ more zbiorek.z.wynikami
Mon Mar 7 21:26:52 MET 1994
$
```

Zgadza się? Jest, co powinno być.

Czasami zechcesz przełączyć strumień z wyjścia jednej komendy na wejście drugiej komendy i stworzyć potok (nie zniechęcaj się, nie robię sobie tylko żartów, potok bardzo się przydaje, np. przy mojej ukochanej funkcji **grep**, o której później[4.2]). Dla uruchomienia potoku połącz komendy symbolem | (pionowa kreska, jest na klawiaturze). Dla przykładu, jeśli zamiast na zbiór (jak wyżej) chcesz wynik polecenia **date** od razu drukować:

```
lksj% date | lpr
lksj%
```

drukowanie

Powinno się wydrukować. Piszę powinno, bo z drukarkami ciężka sprawa. Nie dość, że są sieciowe (na ogół daleko) i nigdy nie wiadomo czy zadziałają to jeszcze jest tysiąc i jeden sposobów drukowania. Zwróć uwagę, że **lpr** działa w C shelu (ten z %) bo w Bourne Shel (ten z $) będzie **lp**. Ale to nie wszystko, lepiej teraz zostawić problem drukowania, przyjdzie czas, że powiem więcej[7.0] (ale i tak z pewnością za mało).

2.5 Uruchamianie zadań w tle

Czasami jest wygodnie uruchomić zadanie z konsoli i, nie czekając na jego zakończenie, mieć możliwość wydawania następnych poleceń. Trzeba to zadanie umieścić w tle. Nic prostszego, wystarczy do polecenia dodać symbol ampersand (**&**) i już:

```
$ duże.zadanie &
[1] 12834
$
```

numer identyfikacyjny

System zapuści zadanie, nada mu numer identyfikacyjny i odeśle do wykonywania w tle, zwalniając powłokę dla następnych poleceń. Jeśli zadanie w tle wysyła

jakieś komunikaty na ekran to niestety mieszają nam trochę w pracy.

Zakończenie pracy w tle jest sygnalizowane przez system dopiero wtedy gdy wykonujesz jakąś komendę a system właśnie skończył zadanie w tle:

```
$ date
Tue Mar 8 11:16:22 MET 1994
[1] + Done duże.zadanie
$
```

Czasami wykonanie zadania będzie trwało tak długo, że nie warto czekać siedząc przed ekranem. Normalne odłączenie się od systemu spowodowałoby zamknięcie zadania wykonywanego w tle. Trzeba użyć polecenia **nohup** przy uruchamianiu zadania:

nie przerywać

```
$ nohup duże.zadanie &
[1] 12836
$
```

po czym możesz się odłączyć od systemu (**exit**) a procesor dalej obrabia Twoje zadanie.

2.6 Pomocy! man

Tyle już wiesz, że warto zdradzić Ci tajemnicę: wcale nie musisz czytać dalej tej głupiej książeczki. Skorzystaj z pomocy zawartej w systemie. Wystarczy, że wydasz polecenie **man** z nazwą komendy jako argumentem i już ekran zapełnia się informacją (jeśli jest zainstalowany na dysku komputera, z którego właśnie korzystasz). W szczególności możesz dostać informację o samym **man**, zwalnia mnie to z obowiązku rozpisywania się na ten temat. Napisz więc:

/usr/man/man?

```
$ man man
Reformating page.  Wait ...  done
```

a ekran zapełni się informacją o tym jak korzystać z tej pomocy (w okienkach spróbuj **xman &**). Zanim to będzie na ekranie warto wiedzieć, że następny ekran wyświetli się po naciśnięciu klawisza spacji.

Jeśli Ci się znudzi to możesz nacisnąć klawisz *Q* i *Enter*, wróci znak zachęty. Warto jednak czytać do końca bo często umieszczane są tam przykłady zastosowań.

Czasami piszesz **man i.coś.tam** a system odpowiada **no manual entry**. Chyba to Twoja radosna twórczość **i.coś.tam** nie jest poleceniem znanym systemowi. Spróbuj wtedy zapytać **apropos** *problem* (ale tylko UNIX V). System będzie próbował podpowiedzieć Ci komendę wypisując komendy, w których opisie występuje wspomniany *problem*. Spróbuj na początek **apropos apropos** lub **apropos who**. Jeśli nadmiar informacji wysyłanej przez system zdenerwuje Cię, nie zawahaj się użyć *Ctrl-C* dla przerwania gadulstwa.

apropos

Jeśli boisz się gadulstwa pomocy typu **man** i **apropos** to jest jeszcze jeden sposób na zwięzłą informację, polecenie **whatis** daje odpowiedź w jednej linii:

```
$ whatis date
date (1) -display or set the date
$
```

Inna sprawa czy to istotnie zmienia Twój pogląd na polecenie `date`. Ten numerek (1) coś znaczy, ale nie chce mi się już rozwijać tematu (to byłoby istotne gdyby Ci przyszła ochota do czytania dokumentacji systemowej, ale nie podejrzewam Cię o to).

To samo polecenie może też dać w wyniku:

```
% whatis date
/usr/man/whatis:  No such file or directory
man:  man entry for date not found
```

co znaczy, niezależnie od tego jak się przetłumaczy, że `whatis` nie działa. Może być system tak skonfigurowany, mogą ścieżki dostępu być niewłaściwe, może czegoś brakuje w `.cshrc` lub parę innych przyczyn (pomijam tu przypadek ewidentnego błędu np. `whaatis` zamiast `whatis`, odpowiedź będzie taka sama, z dokładnością do whaatis). Nie przejmuj się za bardzo. UNIX już taki jest. Na szczęście daje dużo innych możliwości znalezienia pomocy.

Rozdział 3

Manipulacje (zbiorami)

Prawie wszystko jest traktowane w UNIXie jako zbiór (plik) [file], a więc:

- **Dokumenty** – Te, które zawierają raporty, listy, źródła programów lub jakikolwiek inny tekst, który chcesz napisać i zapamiętać.
- **Polecenia** – Większość z nich to zbiory binarne, wykonywalne.
- **Urządzenia [Devices]** – Tak, tak! Terminal, drukarka, napęd dysków traktowane są przez system jak zbiór (pisałem o tym w poprzednim rozdziale[2.3] przy okazji przełączania wyjścia).
- **Katalogi [Directories]** – Katalog to prosty zbiór, który zawiera w sobie inne zbiory.

Nauczmy się manipulować zbiorami.

3.1 Absolutnie podstawowe rzeczy

Przed eksperymentami, do których będę Cię namawiał, sprawdź czy *jesteś w swoim home.* (To niestety żargon, ale każdy do kogo zwrócisz się z jakimś pytaniem dotyczącym zbiorów zaraz będzie ględził o *home.* Trudno, nie znam dobrego polskiego odpowiednika. Czasami używam określenia *katalog użytkownika* ale nie bardzo lubię mówić *Twój katalog użytkownika.*)

Tak, czy inaczej, Twój katalog użytkownika [your home] założył dla Ciebie Szanowny Pan Administrator Systemu przy okazji otwierania konta dla Ciebie. (Trzeba bardzo szanować tego Pana bo on jest Bardzo Ważną Osobą. Jeśli jeszcze o tym nie wiesz to na pewno się przekonasz. Swoją drogą, dlaczego to są przeważnie panowie, owi BWO ?)

Bardzo Ważna Osoba

Wracając do ... *home* to jest to ważna sprawa. Można to sobie wyobrazić jako trochę miejsca na dysku, przeznaczone tylko dla Ciebie, gdzie Ty ustalasz prawa (np. prawa dostępu do zbiorów), kopiujesz, kasujesz, zmieniasz nazwy, itp., itd. ale i Ty odpowiadasz za bałagan. Żeby nie robić bałaganu innym sprawdź więc czy jesteś w swoim home:

```
$ cd
$
```

żadnej informacji ***Bourne shell*** nie daje, ale akcja była. Inaczej przedstawia
powrót do domu się sprawa z *C shell*, ten jest łaskawy i wypisuje Twój *home*:

```
% cd
/home/lks/jsupel
%
```

Gdziekolwiek by Twoja komputerowa dusza nie bujała (w sensie rozległego *drzewa zbiorów*) wrócisz na gałąź Tobie przypisaną czyli Twój *home*. Możesz to sprawdzić komendą **pwd** (to takie pytanie: na której gałęzi drzewa zbiorów jestem?), która wypisze na ekranie nazwę aktualnej *ścieżki dostępu* [path]:

```
$ pwd
/export/home/username
```

Z */export* nie będę Ci się tłumaczył (zresztą nie zawsze się pojawia), ten tutaj **/home** jest tylko po to aby namieszać w tym co wcześniej pisałem o *home* (może jednak lepiej będzie jeśli powiem, że jest to katalog gdzie są umieszczone katalogi użytkowników, więcej nie powiem, bo musiałbym tłumaczyś się z obecności **export** a boję się, że zamieszam) natomiast *username* powinno być Twoją ***nazwą użytkownika*** chyba, że Pan Administrator zrobił to inaczej. (Zwróć uwagę, że pozwoliłem sobie na opuszczenie Szanowny, mogę tak zrobić,
Pan Administrator bo prawdopodobnie chodzi o administratora systemu na lokalnym komputerze, którego łatwiej obłaskawić. Może się nawet okazać, że jest to Miła Pani Ela i można opuścić Pan).

Użyj polecenia **touch** do stworzenia pustego zbioru. Jeśli nazwa jaką mu nadasz jeszcze nie jest w użyciu (nie ma jeszcze zbioru o takiej nazwie) to polecenie **touch** utworzy pusty zbiór. Jeśli taki zbiór istnieje, będzie zmieniona data jego utworzenia (zawartość pozostanie niezmieniona).

```
$ touch tmpzbiór
$
```

Teraz sprawdź poleceniem **ls** czy taki zbiór rzeczywiście został utworzony:

```
$ ls tmpzbiór
tmpzbiór
```

Samo **ls** bez argumentu **tmpzbiór** wypisałoby nazwy wszystkich zbiorów w aktualnym katalogu. Dodanie parameru (nazwy zbioru) powoduje zwrot tego parametru, jeśli zbiór o takiej nazwie istnieje w tym katalogu. Więcej o **ls** bedę pisał później[3.4],[3.5], ale tak na prawdę to zawsze możesz wykonać **man ls** gdzie kawa na ławę.

Użyj polecenia cp do skopiowania zawartości **tmpzbiór** do zbioru o nazwie cpzbiór: *skopiować zbiór*

```
$ cp tmpzbiór cpzbiór
$
```

Teraz spróbuj wylistować oba zbiory. Tak się składa, że nazwy obu kończą się takimi samymi znakami, można więc użyć jokera w miejsce niejednakowych znaków (gwiazdka * jako joker nie precyzuje ile znaków będzię zastąpionych, może być jeden lub żaden lub kilka), jeśli wiemy, że chodzi o jeden tylko znak to jokerem jest ? (znak zapytania). W tym przypadku będzie: *joker ** *joker ?*

```
$ ls *zbiór
cpzbiór tmpzbiór
```

nazwy wypisane są w porządku alfabetycznym. Po szczegóły polecenia cp udaj się do Kolegi Mana (man cp), podobnie o jokerach (gwiazdkach i innych) opowie ci man wildcard lub apropos wildcard *wildcards*

Umiesz już kopiować, więc dowiedz się jak zmienić nazwę:

```
$ mv tmpzbiór innyzbiór
$
```

zniknął tmpzbiór a na jego miejsce pojawił się innyzbiór, o czym łatwo się przekonać wykonując:

```
$ ls *zbiór
cpzbiór innyzbiór
```

Zawartość innyzbiór jest taka sama jaka była w tmpzbiór. Tą drogą (przez mv) możemy także zmienić usytuowanie zbioru w drzewie zbiorów (zmienić ścieżkę dostępu) choć w rzeczywistości niczego nie ruszamy. Poczytaj sobie więcej pisząc man mv.

Trzeba jeszcze umieć usunąć zbiór. Polecenie rm zrobi to:

```
$ rm cpzbiór
$
```

i zostanie tylko: *usunąć zbiór*

```
$ ls *zbiór
innyzbiór
```

Kasowanie zbiorów w UNIXie jest nieodwracalne. Nawet Pan Administrator Systemu nie jest w stanie pomóc (chyba, że właśnie zrobił kopię dysków na taśmę). *kasowanie jest nieodwracalne*

Bardzo Cię proszę uważaj z jokerami (zwłaszcza z gwiazdką) przy kasowaniu plików. Zachce Ci się np. skasować za jednym zamachem wszystkie wyżej utworzone zbiory i wykonać `rm *zbiór` a pomylisz się (ludzka rzecz), napiszesz `rm * zbiór` i nieszczęście gotowe. UNIX, wredna bestia, pomyśli, że chcesz skasować dwie rzeczy: * i `zbiór` (bo zaplątała się spacja za gwiazdką).

pułapki kasowania

A * oznacza wszystkie zbiory w katalogu. UNIX najpierw skasuje wszystkie zbiory, później będzie chciał skasować zbiór `zbiór` i wspaniałomyślnie Cię poinformuje, że nie ma takiego zbioru w Twoim katalagu. Pomyśl dwa razy zanim skasujesz coś z gwiazdką w nazwie, wylistuj używając tej gwiazdki, a później i tak będzie co ma być. Myśl więc także o kopiach zapasowych szczególnie Ci drogich zbiorów.

Jak obejrzeć na ekranie?

Chyba nie muszę już dodawać, że `man rm` może Cię oświecić?

Przypomnę tutaj, że istnieje polecenie **more**, które pozwala obejrzeć na ekranie zawartość zbioru:

```
$ more jakiś.zbiór
...
...
zapełni się cały ekran, jeśli zbiór dostatecznie duży, a na dole będzie
coś takiego:
...
-More- nn% [Pres space to continue, 'q' to quit.]
```

a także polecenie **cat** *jakiś.zbiór*, które też wyrzuca zbiór na ekran, ale czyni to zbyt dosłownie, bez zatrzymania, do końca.

Bardziej ta komenda się przydaje do łączenia zbiorów:

```
$ cat plik1 plik2 plik3 > plikduży
$ ls *plik
plikduży plik1 plik2 plik3
```

jaki zbiór ?

(Zobacz, może ktoś inny napisał coś ciekawszego na ten temat.)

Czasami dobrze jest wiedzieć czy zbiór jest tekstowy czy też binarny (wykonywalny) naprzykład, czy też jeszcze inny. Spróbuj zrobić:

```
$ file jakiś.zbiór
jakiś.zbiór:  ascii text
```

zawsze to coś daje do myślenia. Jeśli już mówimy o zbiorach tekstowych to nie od rzeczy będzie dodać, że isnieje możliwość porównania zawartości dwu zbiorów. Załatwi to za nas polecenie **diff** analizując linia po linii lewy i prawy zbiór:

porównanie zbiorów

```
$ diff lewy prawy
```

Za mało tu miejsca na pokazanie przykładu. Wystarczy, że powiem iż dłuższe zbiory (więcej niż 3500 linii) lepiej sprawdzać programem **bdiff**. Jeśli ktoś chciałby porównywać jednocześnie trzy zbiory, może to zrobić za pomocą **diff3**.

Są jeszcze dwa polecenia kopiowania zbiorów, o których warto pamiętać: *unix2dos*
dos2unix oraz **unix2dos**
dokonują one konwersji zbiorów tekstowych UNIXa z i do DOSu. Różnica w tekstowych zbiorach tych systemów polega na oznaczeniu końca linii. W UNIXie jest to tylko LF [line feed] a w DOSie dwa znaki: LF i CR [carriage return]. Zbiór DOSowy w UNIXie wykaże dodatkowy znak (^M) na końcu każdego wiersza, zbiór UNIXowy w DOSie będzie w jednym wierszu.

Nieco więcej o kopiowaniu zbiorów między systemami powiem przy okazji dyskietek(4.8) i FTP(6.0).

3.2 Katalogi

Pan Administrator założył Ci *konto użytkownika* i przydzielił miejsce na dysku (*home*) lokując je odpowiednio w strukturze drzewa zbiorów (**/home/***username*). Teraz Ty zapełniasz je swoimi zbiorami. Przychodzi czas, że zbiorów jest za dużo, trudno się w tym poruszać. Trzeba zrobić porządek. Zawsze dobrym wyjściem jest skasować zbędne zbiory. Można jednak porządkować zakładając katalogi (podkatalogi katalogu głównego), jednym słowem stworzyć własną strukturę zbiorów. Najpierw poleceniem **mkdir** trzeba utworzyć pusty podkatalog np. z nazwą *teksty*: *założenie katalogu*

```
$ mkdir teksty
$
```

ten zostanie utworzony w gałęzi katalogu, w którm właśnie jesteś (zawsze to można sprawdzić przez **pwd**). Następnie można już otwierać w nim nowe zbiory, bądź kopiować lub przenosić zbiory z innych katalogów. Trzeba tylko pamiętać, że na pełną nazwę zbioru składa się także ścieżka dostępu do zbioru. Najlepiej wyjaśnić na przykładzie: *pełna nazwa zbioru*

```
$ ls
jakiś.tekst teksty
$ cd teksty
$ pwd
/home/username/teksty
$ ls
tekst1 pismo2
$ mv ../jakiś.tekst pismo3
$ ls
tekst1 pismo2 pismo3
$
```

zbiór **jakiś.tekst** został przeniesiony z katalogu głównego do podkatalogu

`teksty/` (zwróć uwagę na skośną kreskę po nazwie katalogu, która odróżnia w ten sposób katalog od zbioru) i zmieniono mu nazwę na `pismo3`. Te dwie kropki w nazwie zbioru przenoszonego poleceniem **mv** oznaczają katalog nadrzędny. Reszta chyba jest jasna.

kopiowanie katalogów

Można kopiować całe katalogi (przeć to są też zbiory!):

```
$ cp -r teksty teksty.nowe
$
```

Ale zamieszanie! Nie dość, że jakieś `-r` to na dokładkę kropka w nazwie katalogu. Kropka w nazwie nie przeszkadza a `-r` nazywa się opcją i zapewnia skopiowanie wszystkich zbiorów zawartych w katalogu `teksty` do nowego katalogu (bez tej opcji polecenie nie byłoby wykonane, system wysłał by komunikat błedu).

usuwanie katalogu

Jeśli można założyć katalog to można też usunąć:

```
$ rmdir teksty
$
```

polecenie **rmdir** działa jeśli katalog jest pusty.

Katalog zawierający zbiory (i podkatalogi) trzeba likwidować jak zbiór, z opcją rekurencji:

```
$ rm -r teksty
$
```

i zamazane zostaną wszystkie zbiory w tym katalogu a także wszystkie zbiory w podkatalogach tego katalogu, a także w podkatalogach podkatalogów tego katalogu, a także... bezpowrotnie.

3.3 Poszukiwanie zbiorów

Czasami trzeba znaleźć zbiór, którego nazwę znamy ale zniknął nam w koronie drzewa katalogów. Jeśli choć w przybliżeniu znamy ścieżkę dostępu możemy listować katalog po katalogu i liczyć na sukces. Da się także zapędzić system do tej roboty. Wystarczy użyć polecenia **find** *katalog opcje*. Tyle tylko, że polecenie jest dość złożone bo wymaga ustawienia parametrów i opcji. Posłużę się najprostszym przykładem:

```
$ find . -name '*k' -print
./plik ./teksty/plik ./teksty/blok ./obrazki/stok
$
```

Zwróć uwagę na kropkę, która tu oznacza katalog roboczy (aktualny) a polecenie **find** interpretuje ją jako: *szukaj zbioru w katalogu bieżącym (roboczym) i jego podkatalogach.* Opcja **name** pozwala nam podać nazwę poszukiwanego

zbioru, w tym przypadku ***k** (i lepiej nazwę podać w nawiasach stringowych), jak widać jokery grają. Na koniec opcja **print** nie wysyła na drukarkę tylko na ekran. (bez tej opcji rezultat działania jest mało interesujący)

Nie muszę mówić, iż innych jeszcze opcji jest tyle, że głowa puchnie. Sprawdź sobie man **find**. Powodzenia.

3.4 Prawa dostępu

Wiele razy wspominałem o tym, że UNIX chroni prywatność użytkownika i sam broni się przed nieuprawnionym działaniem. Jednym z istotniejszych elementów, które dają takie możliwości jest zróżnicowane prawo dostępu do zbioru. Umiesz już listować (wypisywać na ekranie spis) zawartość katalogu poleceniem ls, dodajmy jeszcze opcję *l*. Najpierw jednak zobacz z kim masz do czynienia:

```
$ pwd
/home/lksj/jsupel
```

a teraz jak wygląda lista zawartości katalogu [listing]:

```
$ ls -l
total 8
-rw-r--r-x 1 jsupel 103231 Mar 10 18:26 duży.plik
drwxr-xr-x 2 jsupel   1024 Mar 10 18:22 katalog1
drwxr-xr-x 3 jsupel   1024 Mar 10 18:27 katalog2
-rw-r--r-- 1 jsupel      0 Mar 10 18:23 pusty
```

W każdej linii listy znajdują się:

prawa.dostępu dowiązania właściciel rozmiar data czas nazwa

Pierwszy znak w linii oznacza charakter zbioru. Kreska (-) oznacza zwykły zbiór; literka d oznacza katalog (zbiór zbiorów), inne znaki jeśli się tu pojawią oznaczają specjalne typy zbiorów ale nimi się nie przejmuj (chyba, że musisz).

Następne dziewięć znaków oznacza prawa dostępu do zbioru lub katalogu. Wygląda dziwacznie ale rządzi się jasnymi zasadami i niestety trzeba się w nie zagłębić. Przyswojenie sobie tych zasad uwalnia od wielu kłopotów, zwłaszcza gdy wymieniamy zbiory z innymi użytkownikami systemu.

Z pojęcia dowiązań nie będę Ci się tłumaczył, przyjdzie czas to zapytasz. *właściciel zbioru*

Kolumna z właścicielem jest jasna (póki co).

Dalej już rzeczy mają się właściwie (koń jaki jest każdy widzi).

Dziewięć znaków praw dostępu, trzy grupy po trzy możliwości, taka sobie macierz 3x3 (ale tak naprawdę 3x4, zaraz się przekonasz).

u [user] - pierwsza grupa znaków - prawa użytkownika

g [group] - druga grupa znaków - prawa grupy

o [others] - trzecia grupa znaków - prawa pozostałych

a [all] - wszystkie grupy razem

a wewnątrz grupy znaków:

r [read] można czytać

w [write] można mazać (dopisać, usunąć)

x [execute] można wykonać

\- nie można {czytać, mazać, wykonać}

Kto ustala te prawa? System oczywiście. Na początek jest:
`-rw-r--r--` dla zbiorów, `drwxr-xr-x` dla katalogów,
później system pozwala Ci zmienić te prawa ale tylko w sosunku do zbiorów, których jesteś właścicielem (kto jest właścicielem informuje przy okazji `ls -l`).

zmiana prawa dostępu

Jak zmienić prawa dostępu? A przez polecenie **chmod**. Powiedzmy, że jest tak:

```
$ ls -l pusty
-rw-r--r-- 1 jsupel 0 Mar 10 18:23 pusty
```

zbiór pusty ma ograniczone prawa mazania, tylko Ty możesz mazać. Chcesz dodać to prawo wszystkim? Nic prostszego, chmod to załatwi:

```
$ chmod a+w pusty
$ ls -l pusty
-rw-rw-rw- 1 jsupel 0 Mar 10 18:23 pusty
$
```

Proste? Te trzy znaki znaczą: (a)wszystkim (+)dodać (w)mazanie. Jeśli chcesz odjąć uprawnienia to ten kawałek będzie wyglądał np. tak:
`o-r` (o)pozostałym (-)odbieram (r)możliwość czytania.

Są jeszcze inne sposoby użycia **chmod** ale zostaw je Szanownemu Panu Administratorowi (on może wszystko i każdemu dodać, ale i odjąć, taki On mocny).

3.5 Odkryj ukryte (ls-a)

zbiory z kropką

Jest kila zbiorów, które nie są wykazywane przy zwykłym ls. Pewnie dlatego są ukryte, że są w nich tajemne rzeczy i zwykły śmiertelnik nie powinien ich nawet oglądać. Po części jest to prawdą, ale przyjdzie czas[9.1], że trzeba będzie coś z nimi (w nich) zrobić. Czasami też ktoś Ci powie: *zajrzyj do swojego* .cshrc i najdzie Cię zdrożna ochota podejrzeć. Możesz przekonać się czy masz takie zbiory dodając do ls opcję -a. Wylistuje to wszystkie, naprawdę wszystkie zbiory w bieżącym katalogu. Nie obawiaj się, że od wylistowania coś się zepsuje. Nie daję przykładu bo nie. Może później.

A może już dosyć o zbiorach? Chyba jednak dam Ci ściągawkę.

Polecenia manipulacji zbiorami	
Polecenie	Co się dzieje?
cd	zmieniasz katalog
pwd	dowiesz się gdzie jesteś
mkdir	otwierasz nowy katalog
rmdir	usuwasz katalog
ls	lista zbiorów w katalogu
ls -a	lista wszystkich zbiorów
ls -l	lista (pełna) zbiorów
file	pokaże typ zbioru
cat	wyświetli zawartość zbioru
cp	kopiuje zbiory
mv	przesuń lub zmień nazwę
rm	usuń zbiór
touch	kreuje zbiór
chmod	ustala prawa dostępu
lp	drukuje zbiór (Bourne shell)
lpr	drukuje zbiór (C shell)
lpq	sprawdza drukowanie (C shell)
lprm	usuwa zadanie drukowania (C shell)
dos2unix	kopiuje pliki tekstowe z DOSa
unix2dos	kopiuje pliki tekstowe do DOSa

Jeszcze raz o kasowaniu zbiorów

Wiemy już, że kasowane(3.1) zbiory są nie do odzyskania.

Czy jednak na pewno? Może to tylko sprawa "biegłości", a zawartość zbioru można przeczytać bezpośrednio z dysku, bez pomocy funkcji systemowych (jak w dobrym, poczciwym DOS)?

Niestety jest to praktycznie niemożliwe. Teoretycznie można sobie wyobrazić taką sytuację, że natychmiast po mylnym skasowaniu zbioru odmontowywany jest dysk i coś tam robimy. Tyle tylko, że nikt się tego nie podejmie. Odłączenie dysku, w systemie wielodostępnym, nie jest sprawą prostą ani miłą.

Pozostaje zabezpieczenie się przed nieumyślnym skasowaniem pliku. Pomocną będzie tu opcja `-i` polecenia **rm** wymuszająca interaktywny styl pracy. Można na stałe wpisać do zbioru `.cshrc`(9.1) lub `.profile`(9.4) alias polecenia **rm** o postaci:

```
alias rm    'rm -i'
```

co zmusza do potwierdzenia każdej operacji kasowania.

Innym rozwiązaniem jest instytucja "śmietnika" - katalogu czasowo przechowującym kasowane zbiory (zastosowanym np. w systemie okien OpenLook). Zamiast kasowania **rm** zbiór jest przesuwany **mv** do pośredniego katalogu (śmietnika) i dopiero stamtąd usuwany przy okazji robienia porządków.

Rozdział 4

Trochę więcej (wiedzieć)

Nie zaszkodzi wiedzieć o innych poleceniach ułatwiających życie a czasami wręcz koniecznych lub często używanych.

4.1 Zmiana hasła

Dla bezpieczeństwa systemu wymagane jest hasło dla każdego użytkownika. Zmiana hasła kilka razy do roku da Ci poczucie dobrze spełnionego obowiązku: nie będziesz tym, który ułatwia włamanie do systemu. Jeśli zauważysz, że ktoś pracuje na twoje konto, bez Twojego zezwolenia, zmień natychmiast hasło. Polecenie passwd załatwia sprawę:

```
$ passwd
Changing password for jsupel on lksj
Old password:
New password:
Retype new password:
$
```

Zastosuj się do próśb i wpisuj kolejno stare i dwa razy nowe hasło. System zachowa się dyskretnie i nie powiela na ekranie wpisywanych haseł. Hasło nie może być krótsze niż sześć znaków. Długość dowolna, choć system bierze pod uwagę osiem pierwszych znaków. Hasło nie powinno być słownikowe (żaden kłopot dla włamywacza "przepuścić" hasło przez słownik). Najlepiej jeśli hasło zawiera litery (duże i małe) i cyfry. Same duże litery – niedobrze. Same cyfry – jeszcze gorzej. Administrator systemu może ustalić obowiązujący czas co jaki trzeba zmieniać hasło. Jeśli nie zmusza Cię do tego to znaczy, że jest litościwy lub (co bardziej prawdopodobne) obawia się, że przyciśnięty obowiązkiem ciągłych zmian zaczniesz zapisywać hasło na kartce i trzymać w pobliżu terminala. To byłby koniec.

minimum sześć znaków

4.2 grep

przeszukiwanie zawartości zbiorów

Jest to bardzo pożyteczna funkcja. Pozwala znaleźć dowolny ciąg znaków w różnych sytuacjach. Najprościej użyć **grep** *ciąg.znaków zbiór* (gdzie ciąg znaków może być słowem lub frazą) do przeszukiwania zbiorów. Odpowiedzią będzie wypisanie całej linii tekstu zawierającą wskazany ciąg znaków.

Bardziej zaawansowany sposób to używanie **grep**, w połaczeniu z innymi poleceniami systemowymi, jako filtra. W tym celu trzeba skierować wyjście z polecenia do **grep** (stworzyć potok):

```
$ ls -l *.ps | grep May
-rw-r--r-- 1 jsupel 12732 May 24 12:13 cloc.ps
```

po wykonaniu pełnego listingu zbiorów typu *.ps w aktualnym katalogu (`ls -l *.ps`) wynik zostaje przekazany do **grep**, ten zaś zostawi tylko te linie, które zawierają ciąg znaków **May**.

Jeśli poszukiwanym ciągiem znaków jest fraza (zawiera spacje) to lepiej ten ciąg zamknąć w nawiasach stringowych (pazurkach lub ząbkach). Więcej szczegułów znajdziesz oczywiście przez **man grep** ale ja dodam, że możesz poszukiwać także znaków specjalnych, znaków końca lub początku linii a nawet negacji znaku (tzn. wszystkie linie, które nie zawierają wyszczególnionego znaku).

4.3 Zabić [kill]

Grożnie brzmi ale czasami jakieś działanie komputera doprowadzi Cię do takiej furii, że nic tylko zabić! System pozwala utłuc niepokorny proces. Najpierw musisz jednak wiedzieć co (na szczęście nie kogo) chcesz zabić.

Po zinterpretowaniu przez system każdego polecenia uruchamiany jest *proces*, z unikalnym numerem identyfikacyjnym [PID], istniejący aż do wykonania polecenia. System używa PID do śledzenia bieżącego stanu każdego procesu.

wykaz aktualnych procesów

Polecenie **ps** informuje o aktualnie istniejących procesach a także o innych sprawach jak: z którego terminala uruchomiony (istotne w systemie okienek), ile czasu trwa, ile czasu procesora zajmuje itp. (odsyłam jak zwykle do **man ps**). Podaje nawet nazwę polecenia, które go uruchomiło. Z tej informacji już da się skorzystać, bo powiążesz numer PID z poleceniem, które chcesz utłuc. Zabija się proces poleceniem **kill** *PID*, stąd konieczność znania numeru procesu. Przykład zabijania procesu:

```
$ ps
PID    TTY    TIME    COMMAND
1291   co     0:12    -bin/csh  (csh)
3250   p0     0:00    ps
1286   p1     0:05    -bin/csh  (csh)
3244   p1     0:05    vi tekst
```

```
$ kill 1291
[1]   Terminated          -bin/csh/  (csh)
$
```

Czasami proces jest oporny, nie poddaje się tak łatwo. Przyrzuć go wtedy dziewiątką (9):

```
$ kill -9 PID
```

i już (oczywiście *PID* to numer procesu, który Ci dopiekł).

Polecenie **ps** ma oczywiście swoje opcje, do sprawdzenia **man ps**, z których b.użyteczna jest (-u) bo listuje wszystkie procesy, których właścicielem jest *użytkownik* (np. **ps -u jsupel**).

4.4 Praca w sieci

Sieciowe połączenie między komputerami pozwala na przesyłanie informacji między komputerami. Jeśli mówimy o sieci to najczęściej mamy na myśli sieć lokalną, ograniczoną do budynku lub np. terenu uczelni. Sieci lokalne [Local Area Network] (LAN) mogą łączyć się w większe [Wide Area Network] (WAN) lub, co obecnie realizuje się w Polsce, w sieci miejskie [Metropolitan Area Network] (MAN) skąd nazwa powstającej sieci w obrębie warszawy WARMAN.

Trudno, zwykłemu użytkownikowi, określić granice sieci. W pierwszej kolejności będzie to kryterium terytorialne ([area]), następnie związane z systemem obsługującym sieć (np. sieć PCtów z systemem Nowell), dalej z protokołem transmisji sieciowej (Internet, Bitnet, Decnet itd.). Istnieją wyspecjalizowane komputery ([gateway]), które umożliwiają przepływ informacji między różnymi sieciami. W końcu można przyjąć, że cały świat "jest w sieci" WWW [World Wide Web].

World Wide Web

Dla naszych dalszych prób przyjmijmy, że siecią jest sieć lokalna (LAN), w której istnieje baza danych o elementach sieci zarządzana centralnie przez NIS [Network Information System].

4.4.1 rlogin

Będąc już zalogowanym w systemie (brr... jaki żargon!) lub innymi słowy jeśli już masz otwartą sesję, możesz otworzyć następną sesję na innej stacji UNIXowej (zwróć uwagę, na podkreślenie, że chodzi o komputer z systemem UNIX a nie PC z DOSem):

```
pierwszy% rlogin drugi
```

gdzie *drugi* jest nazwą innej (oddalonej) maszyny. Jeśli masz założone konto użytkownika na oddalonej maszynie, lub zadziała **automonter** - program, który wykona pewne prace po zasięgnięciu danych z bazy NIS, to będziesz

odnosić wrażenie, że jeszcze raz zaczynasz sesję[1.1] (i tak jest w rzeczywistości), system zapyta Cię o hasło a następnie wyświetli blok informacji i znak zachęty [prompt]. Możesz dalej pracować jak na poprzednim komputerze, korzystając z tego samego otoczenia i zmiennych systemowych (jeśli Twój *home* został przyłączony przez **automonter**) inaczej mówiąc nie ma żadnej różnicy (może poza szybkością działania komputera).

Sytuacja jest więc taka, że Twoja pierwsza sesja jest jakby zawieszona (odnosi się to tylko do sytuacji pracy z terminalem alfanumerycznym, przy pracy w "oknach" mamy tyle terminali ile otwartych okien) a czynną jest nowa sesja. W szczególności można by teraz wykonać `rlogin` do komputera przy którym siedzimy - *pierwszy* (system to wykona) tylko po co?

Inna sprawa, gdy na komputerze *drugi* nie masz *home*. Wtedy też możesz otworzyć sesję ale system poinformuje Cię o przyłączeniu do katalogu głównego (`root`): ....

```
No directory! Logging in with home=/
drugi%
```

brak home

konsekwencją jest brak dostępu do własnych zbiorów, a w szczególności do zbiorów konfiguracyjnych, wśród nich do `.cshrc`. Będziesz więc odnosić wrażenie, że "wszystko jest inaczej" i coś co przedtem "chodziło" teraz "nie chodzi".

Lepiej w takiej sytuacji wykonać:

```
drugi% logout

Connection closed
pierwszy%
```

i wrócić do pierwszego komputera. Zamknięcie sesji jest więc typowe i ważne jet też to co zostało już wcześniej[1.2] powiedziane na ten temat. Czasami, korzystając z sesji otwartej przez jednego użytkownika, chcesz otworzyć sesję innego użytkownika. Wtedy do polecenia `rlogin` należy dodać parametry:

```
komputer1% rlogin -l inny.użytkownik komputer2
```

dalej będzie pytanie o hasło innego użytkownika i otwarcie dla niego nowej sesji. W szczególności zamiast `komputer2` może być `komputer1` a zamiast *inny.użytkownik* może być *mój.uid*.

Jeśli już pogubisz się w tych zdalnych sesjach zawsze możesz dowiedzieć się od systemu kim jesteś (ładnie brzmi prawda?):

```
% who am i
```

odpowiedź bywa prozaiczna, niestety. Okazuje się, że nie możesz komputera zadziwić retorycznym: "Pan nie wie kim ja jestem!".

4.4.2 `telnet`

Innym programem do otwierania sesji roboczej na zdalnym komputerze jest program `telnet`. Działa tak samo jak `rlogin` (z dokładnością do wiedzy i umiejętności Administratora Systemu). Wywołuje się go:

```
% telnet nazwa.komputera
```

i dalej już standardowo: podaje się nazwę użytkownika i hasło a system wyświetla blok komunikatów i znak zachęty.

Tym co odróżnia `telnet` od `rlogin` jest po pierwsze możliwość podania adresu zdalnego komputera w pełnej, internetowej, postaci[5.1] co umożliwia otwarcie sesji nawet na bardzo odległym komputerze (decyduje tu szybkość łączy, ale powszechną praktyką jest korzystanie z komputerów w USA, jeśli tylko ma się dostęp do konta), po drugie `telnet` istnieje także w postaci DOS-owej, możliwe jest więc otwarcie, na odległym komputerze UNIXowym, sesji UNIXowej z PCtem jako terminalem (warunkiem jest oczywiście włączenie PCta do sieci). *zalety telneta*

Pewną namiastką zdalnego otwierania sesji jest polecenie `rsh`. Umożliwia ono wykonanie jenej komendy na drugim komputerze sieciowym, bez konieczniści otwierania sesji:

```
komputer1% rsh komputer2 polecenie
```

oczywiście nie wszystko da się w ten sposób zrobić, ale np. można sprawdzić zawartość jakiegoś katalogu itp.

4.4.3 Kopiowanie między komputerami

Obecność w sieci umożliwia też kopiowanie zbiorów z innego komputera (pod warunkiem, że są dla Ciebie dostępne[3.4]) poleceniem `rcp`. Wyglądać to może naprzykład tak:

```
komputer1% rcp komputer2:/home/janek/list1 .
komputer1%
```

Zwróć uwagę na dwukropek po nazwie zdalnego komputera oraz na kropkę na końcu. Ta kropka oznacza, że skopiujesz plik `list1` do bieżącego katalogu (zamiast kropki może być oczywiście pełna nazwa, ze ścieżką). Gdybyś chciał skopiować cały katalog (wraz z plikami, które zawiera) to wstaw opcję (r) (tzn. `rcp -r komputer2:/ho.....`). kopiowanie jest możliwe oczywiście w obu kierunkach (ze swojego konta możesz komuś posłać zbiór, musisz tylko znać jego *home*). *rcp*

Zamiast `rcp` polecam jednak, jako użyteczniejszy, program FTP[6.0]. Różnica między `rcp` a FTP jest podobna jak między `rsh` a `telnet`.

4.4.4 Kto też?

who

Ponieważ system jest wielodostępny, prawdopodobnie jeszcze ktoś inny, równocześnie z Tobą, dzieli czes komputera. Polecenie **who** pokazuje aktualnych użytkowników zarejestrowanych w systemie:

```
% who
jsupel ttyp2 Apr 12 11:50 (lkpc2)
.....
```

Czasami wystarczy tylko **w** aby otrzymać te same informacje.

rusers

Jeśli Twój komputer (terminal) włączony jest do sieci możesz dowiedzieć się kto w tej chwili jest przyłączony do systemu i jakiego używa komputera. Załatwia się to poleceniem **rusers**, dobrze jest dodać opcję **-l**:

```
% rusers -l
ablin lks:ttyp1 Apr 12 11:41 4 (psa5pc:0.0)
wsosn zmit2:con Apr 12 11:16
.....
```

co by oznaczało, że użytkownik *ablin* używa pierwszej konsoli (ttyp1) komputera o nazwie *lks* od godz. 11:14 12 Kwietnia, od 4 minut nie wydawał poleceń, a tak wogóle to siedzi przed terminalem o nazwie *psa5pc* i pracuje w okienkach (:0.0). Natomiast użytkownik *wsosn* właśnie pracuje przy komputerze *zmit2* itd.

Podobnie interesującym może być uzyskanie informacji o komputerach włączonych do sieci. Polecenie **rup** listuje wszystkie czynne komputery i podaje informację o ich zajętości (obciążeniu):

```
% rup
lks  up 38 days, 20:23, load average:  1.38, 1.35, 1.36
lkb3 up 17 days, 21:38, load average:  0.12, 0.03, 0.00
....  dalej następne maszyny
```

Stąd można wybrać mniej obciążony komputer jeśli zależałoby Ci na szybszym wykonaniu zadania.

Grupa trzech liczb na końcu wiersza oznacza średnie obciążeie w przedziale ostatnich: 5 sek, 1 min., 10 min. Można więc odczytać, że maszyna *lks* wykonuje jakieś dłuższe zadanie i jest dociążona (wskaźnik load average = 1.00 świadczyłby o średnim obciążeniu) natomiast maszyna *lkb3* wykonuje właśnie jakieś zadanie ale poprzednio nie była dociążona (po szczegóły określania obciążenia odsyłam, jak zwykle, do **man rup**).

finger

Informację o użytkownikach można także uzyskać poleceniem **finger**. Użyte bez opcji i argumentów zadziała tak samo jak **who** lub **w**. Jednak dużo bardziej użyteczne jest to polecenie dla uzyskania szerszej informacji o użytkowniku. Jeśli argumentem polecenia **finger** będzie adres sieciowy (niekoniecznie lokalny) to system roześle "wici" po sieci i wyszuka dane o interesującym nas użytkowniku. Wygląda to mniej więcej tak:

```
% finger jsupel@ippt.gov.pl

Login name: jsupel              In real life: Jerzy Supel
Directory: /home/lks/jsupel     shell: /bin/csh
On since Apr 12 14:07:08 on pts/0 from lkpc2
5 minutes 59 seconds
Mail last read Tue Apr 12 13:59:23 1994
No plan
```

Jeśli maszyna jest w sieci, można użyć polecenia finger do sprawdzenia innej maszyny w sieci, należy napisać po znaku @ nazwę maszyny, którą chce się sprawdzić:

```
$ finger @lks
[lks]
No one logged on
```

i okaże się, że maszyna *lks* nie jest zbyt zajęta.

4.4.5 Plotkowanie z innymi

Gdy już wiesz kto jest aktualnie zarejestrowany w systemie może zachcesz wysłać mu jakąś wiadomość. Polecenie talk umożliwia połączenie z drugim komputerem (terminalem) i wymianę informacji w czasie rzeczywistym. Wywołanie polecenia talk *nazwa.użytkownika@maszyna* *talk*

```
$ talk jkowal@lks
```

spowoduje przykrycie aktualnego ekranu nowym, podzielonym na dwie części (kreską w połowie wysokości ekranu). W górnej części pojawi się informacja:

```
[Waiting for your party to respond]
```

wyświetlana do czasu uzyskania połączenia.

Natomiast na ekranie wywoływanej osoby ukazuje się komunikat:

```
Message from Talk_Daemon@lks at 10:12 ...
talk:  connection requested by jsupel@lksj
talk:  respond with:  talk jsupel@lksj
```

Jeśli chcesz odpowiedzieć napisz polecenie talk *nazwa@użytkownika* tak jak to zostało zasugerowane. Podczas gdy talk jest wykonywany ekran zostaje podzielony na dwie części i to co piszesz pojawia się w górnej części ekranu, natomiast to, co pisze drugi użytkownik, w dolnej części. Sygnałem do przerwania połączenia jest *Ctrl C*.

Nadużywanie przesyłania wiadomości w czasie rzeczywistym (`talk`) nie jest wskazane, zwłaszcza przy połączeniach między odległymi komputerami. Zabiera czas komputera i blokuje łącza. Dodatkowo może być przyczyną "zawieszenia" połączenia (nie tylko naszego ale wszystkich).

4.4.6 Wiadomości systemowe: `news`

Systemy UNIX mają wbudowane polecenie wyświetlające pliki z informacjami ciekawymi dla użytkowników. Pliki są tworzone przez administratora systemu i umieszczane w specjalnym katalogu. Po umieszczeniu w tym katalogu (najczęściej `/usr/news` pliki stają się *wiadomościami* dostępnymi wszystkim użytkownikom.

W celu przeczytania wiadomości systemowych używamy polecenia `news`, które podane bez opcji wyświetla informacje jeszcze nie przeczytane.

4.5 Ile miejsca na dysku?

Niezależnie od tego jak duże są dyski i ile ich jest w systemie, zawsze brakuje na nich miejsca. Taki już UNIX jest. Społeczność (inni użytkownicy razem z administratorem) oczekuje, że skasujesz swoje zbiory, najlepiej wszystkie. Wtedy jest więcej miejsca dla innych. Nie na długo oczywiście. Zdrowy egoizm nakazuje jednak trzymanie pewnej liczby zbiorów niezbędnych do pracy. Poza tym nie musimy być bardziej "wyrywni" od innych i możemy kierować się zdrowym rozsądkiem. Zdrowy egoizm w połączeniu ze zdrowym rozsądkiem, to jest to! Trzeba tylko orientować się ile miejsca na dysku zajmujemy. Pomoże w tym polecenie `du`. Zapuszczone nawet bez parametrów, podaje ilość miejsca wykorzystywanego przez Twoje zbiory w każdym katalogu i podkatalogach. Dla bardziej wnikliwych, tych którzy chcą zobaczyć jak wyglądają na tle całości, użyteczym będzie polecenie `df`, które podaje informację o wszystkich dyskach w systemie. Ten dysk, który bezpośrednio Cię może interesować będzie miał w nazwie początek nazwy Twojego katalogu (tego, który wyświetla polecenie `pwd`) ale nie pogniewaj się jeśli to będzie tylko `/home`.

Przy pracy z edytorami tekstu (i nie tylko) powstają zbiory przejściowe, które system sam zakłada, bez Twojej akceptacji. Po zakończeniu pracy nie zawsze są one kasowane i zostają w katalogu `/tmp`, biedne sieroty, na dokładkę Ty jesteś ich właścicielem. Zwykłe przeglądanie własnych katalogów nie ujawni tych zbiorów, trzeba ich szukać w katalogu `/tmp` i stamtąd usuwać. Pomocnym może tu być mały skrypt (program napisany w języku powłoki) automatycznie kasujący te zbiory po zakończeniu pracy:

kasowanie /tmp

```
unalias rm
cd /tmp
rm -rf *.*
```

Ten ciąg poleceń (skrypt) może być wpisany do zbioru `.logout`[9.3] który jest automatycznie uruchamiany na zakończenie Twojej sesji.

Największym pożeraczem miejsca na dysku jest zbiór `core`. Ta bomba (Ci, którzy używają okienek widzą ikonę z czerwoną bombką) jest zrzucana przez system w przypadku awaryjnego zatrzymania Twojej sesji. Zawiera w sobie mnóstwo informacji, całkowicie niepotrzebnych dla nowicjusza. Ten zbiór może być (a nawet powinien być) kasowany (`rm core`).

4.6 Jak używać dyskietek?

Dyskietki $5\frac{1}{4}$ i $3\frac{1}{2}$ cala to są fanaberie przyniesione z PCeta. Prawdziwa unixowa stacja robocza to napęd taśmy 1/2 cala i pojemności 150 MB albo Exabajt o pojemności 2.4 GB a jeszcze lepiej 5 GigaBajtów, w ostateczności może być, modny ostatnio, CD ROM. Jasne więc, że z dyskietkami są kłopoty. Pół biedy jeśli używasz PC, który jest "napędzany" unixopodobnym systemem, tam "jakoś" dyskietki będą widziane przez system. Prawdziwa stacja, aczkolwiek może być wyposażona w napęd dyskietek $3\frac{1}{2}$ cala (jak np. SPARC 2) to sama z siebie nie zobaczy dyskietki wsuniętej do kieszeni napędu. Trzeba wykonać kilka magicznych czynności, zarezerwowanych wyłącznie dla osoby znającej hasło `root`. Jednym słowem, trzeba angażować, do skopiowania z dyskietki jednego małego zbioru, Bardzo Ważną Osobę. A to wiadomo jak się kończy. Są jednak osoby odważne i prawdopodobnie ich liczne ataki na BWO doprowadziły do tego, że w niektórych systemach (jak np. w SUN IPPT) działają pewne polecenia umożliwiające zwykłemu śmiertelnikowi skopiowanie zbiorów na (lub z) dyskietkę.

Załóżmy, że masz dyskietkę 3.5" sformatowaną w sposób właściwy dla DOS z gęstością zapisu 1.44 MB a na niej zbiory zapisane pod DOSem. Jeśli w komputerze jest napęd dyskietek (napędy dyskietek mają np. Sparc2 lub Sparc IPC) to włóż dyskietkę do kieszeni napędu. Zwróć uwagę, że napęd nie ma przycisku, który można przycisnąć aby wyskoczyła dyskietka. O wyjmowaniu dyskietki za chwilę. Teraz wpisz polecenie (jeśli siedzisz przed maszyną *lksj* to dokładnie to samo, jeśli jest to inna maszyna to zastąp *lksj* nazwą tej maszyny):

pcmount

```
lksj% rlogin -l pcmount lksj
...
lksj%
```

Reakcja na to polecenie jest dość złożona, nie będę tu wszystkiego przytaczał. Jak widzisz z polecenia, jest to przyłączanie fikcyjnego użytkownika o nazwie *pcmount*. Na końcu jednak wszystko wraca do stanu wyjściowego (znów jesteś użytkownikiem). Skutek tego polecenia jest taki, że pojawił się katalog `/pcfs`, w którym znajdziesz zbiory zapisane na dyskietce. Wystarczy więc wykonać:

```
lksj% cd /pcfs
lksj% ls -l
```

aby wylistować zawartość dyskietki.

Jeśli już jesteś w katalogu **/pcfs** to zbiory DOSowe kopiuj z dyskietki np. tak:

```
lksj% dos2unix zbiór.dos ~/zbiór.unix
lksj%
```

dos2unix
unix2dos

pamiętając, że **~/** oznacza Twój główny katalog [home].

W kierunku przeciwnym, z UNIXa do DOSu można kopiować tak:

```
lksj% unix2dos plik.unixowy /pcfs/plik.txt
lksj%
```

jeśli akurat jesteś w innym katalogu niż **pcfs**. Pamiętaj też, że nazwa zbioru w DOS jest ograniczonej długości.

Możesz teraz wyjąć dyskietkę dając polecenie:

```
lksj% eject
lksj%
```

i włożyć następną i też wykonywać polecenia kopiowania (i inne). Dopiero polecenie odmontowania napędu dyskietek:

```
lksj% rlogin -l pcumount lksj
...
lksj%
```

spowoduje odłączenie katalogu **/pcfs** i zakończenie przygody z dyskietkami. Pamiętaj o odmontowaniu dyskietek, gdyż pozostawienie zamontowanego katalogu **/pcfs** jest przyczyną wielu niezrozumiałych dla Ciebie komunikatów systemowych i źródłem stresów.

> *Uwaga! Polecenie* **eject** *zawsze zadziała, nawet jeśli dyskietki są niezamontowane (jeszcze nie lub już nie).*

flmount
flumount

Dla dyskietek zaformatowanych "pod Unixem" montowanie i odmontowanie wykonuje się poleceniami **flmount flumount**.

Nie będę tu omawiał kopiowania zbiorów wewnątrz programu PC/NFS jako że wykracza on poza ramy tego opracowania. Ci, którzy mają zainstalowany PC/NFS mają też instrukcję użytkownika [user manual].

O innych sposobach kopiowania zbiorów między systemami (a tym samym na dyskietkę) powiem przy okazji omawiania FTP[6.0] [File Transfer Protocol].

Rozdział 5

Poczta (E-mail)

Czy powinienem zdefiniować pojęcie poczty elektronicznej? Pewnie każdy coś już słyszał i wie. Może jednak? Poczta elektroniczna jest to zespół urządzeń informatycznych i telekomunikacyjnych oraz systemów i programów komputerowych, który umożliwia użytkownikowi komputera wysyłanie i przyjmowanie listów. O zasięgu poczty decyduje przyłączenie komputera do sieci komputerowej. Jeśli jest to sieć Internet - łączność jest praktycznie z całym światem.

Nie będę się tu rozpisywał o armii ludzi, którzy czuwają nad poprawnym działaniem poczty ani też o cudach techniki satelitarnej łączności. Nas interesuje program na lokalnym komputerze, który ułatwi przeczytanie, napisanie, wysłanie, otrzymanie, przechowanie lub skasowanie listu. Załatwia to jeden program - `mail`.

Tak naprawdę to przesyłaniem poczty między komputerami zajmują się całe (pod)systemy. Podstawowym w UNIXie jest **UUCP** i stosowany w nim do obsługi poczty program `smail`. Częściej używanym jest *Internet* z podstawowym programem obsługi poczty `sendmail`. Działanie tych programów zachodzi poza zasięgim zwykłego użytkownika i możesz nic o nich nie wiedzieć. Tych kilka nazw podałem tylko dla "osłuchania się", być może kiedyś je usłyszysz i dobrze będzie jeśli skojażą Ci się z pocztą elektroniczną.

UUCP

sendmail

I od razu, na początku, pewien szkopuł. Otóż jest kilka programów UNIXowych obsługujących pocztę elektroniczną. Działają podobnie, tzn. skutek ogólny jest podobny: ułatwiają przesyłanie (wysyłanie i przyjmowanie) krótkich (względnie) tekstów między użytkownikami komputerów połączonych siecią. Różnice miedzy programami obsługi poczty są jednak na tyle istotne, że opiszę oddzielnie dwa najbardziej popularne programy. W UNIXie BSD (np. SunOS 4.1 lub Solaris 1.0) jest to program `Mail`. W UNIXie Systemu V (np. SVR4 lub Solaris 2.0) jest to program `mailx`. Na dokładkę jest jeszcze program `mail`, który działa w obu systemach (ale w różny sposób oczywiście), a wywołanie `Mail` w systemie SVR4 uruchamia program `mailx`. Dosyć aby się zaraz zagubić i mieć pretensje do siebie i świata, że wczoraj coś działało a dziś nie chce.

Szczęśliwi, którzy "pracują w okienkach". Tam mogą się różnić tylko kolejne wersje okienek, poczta działa tak samo (mniej więcej).

5.1 Adresy pocztowe

Adres pocztowy w systemie UNIXowym zawiera informacje konieczne do zidentyfikowania indywidualnego użytkownika. W odróżnieniu od adresu zwykłej poczty adres w UNIXie stanowi pojedyńczy łańcuch znaków. Składa się z dwu części: identyfikatora użytkownika i pełnej nazwy docelowej komputera (*domeny*). Ogólna postać adresu jest taka:

identyfikator@domena

Ucho słonia **@** lub małpa, jak niektórzy wolą, jest integralną częścią adresu. Niektórzy nawet odróżniają adres *Internetu* od innych adresów po tym właśnie znaczku (choć nie do końca jest to prawdą).

Mój adres poczty elektronicznej wygląda tak:

domena `jsupel@lksu.ippt.gov.pl`

Jak łatwo zauważyć ta domena składa się z czterech części oddzielonych kropkami. Aby ją rozszyfrować czytajmy od prawej do lewej.

`pl` oznacza Polskę. Nie jest to obowiązującą regułą, ale przyjętym zwyczajem, że dwie ostatnie literki oznaczają kraj. Łatwo rozszyfrować: uk - United Kingdom, fr - Francja, be - Belgia itd. USA jako reszta świata nie używa tego członu.

.gov. Przyrostek `gov` zwyczajowo oznacza instytucję rządową. Może być też `edu` (uczelnie), `mil` (military), `com` (instytucje handlowe) lub jakiś inny, praktycznie dowolny, ale jednak podlegający zatwierdzeniu w *NIC* [Network Information Center].

Następny człon jest już zwykle nazwą sieci lokanej. W tym przypadku jest skrótową nazwą Instytutu (`IPPT`) .

Pierwszy człon na prawo od małpy jest nazwą komputera, który obsługuje skrzynkę pocztową. W sieciach, gdzie funkcjonuje baza danych o użytkownikach (np. NIS [Network Information System]) nie jest koniecznym podawanie w adresie nazwy maszyny. Wysyłając pocztę w świat system sam dopisze do adresu zwrotnego odpowiedni człon.

5.2 `Mail` w UNIXie BSD

Są dwa podstawowe tryby działania programu pocztowego `mail`: tryb wysyłania poczty [input mode] i tryb przeglądania [command mode] otrzymanej poczty. Tryby pracy różnią się na tyle, że będziemy omawiać je oddzielnie. Zacznijmy od tego co wydaje się łatwiejsze.

5.2.1 Przeglądanie poczty

Przyjmimy, na czas nauki, że nazywasz się Jan Kowalski, a Twój UID (nazwa użytkownika[1.4]) jest `jkowal`, komputer, do którego chcesz się przyłączyć nazywa sie `wawa` i będziesz pracował w powłoce[1.5] C (`csh`). Jeśli już masz za sobą I-szy rozdział to umiesz rozpocząć sesję. Ponieważ są różne sposoby

otwierania sesji[1.3], przyjmijmy, że już jest otwarta. Sprawdźmy, na wszelki wypadek, kto jest użytkownikiem:

```
wawa% whoami
jkowal
```

Twój katalog użytkownika (*home*[3.1]), który sprawdzisz przez:

```
wawa% pwd
/home/wawa/janek
```

jest, na razie, pusty. Sprawdzisz to przez wylistowanie[3.1] zbiorów:

```
wawa% ls
wawa%
```

Oczywiście pusty katalog nie jest warunkiem koniecznym do rozpoczęcia pracy z programem `mail`. Zaczynamy od początku aby łatwiej śledzić zmiany w katalogu, zależne od dalszego działania.

zapuszczamy pocztę

Zacznijmy w końcu pracę z programem poczty elektronicznej:

```
wawa% Mail
No mail for jkowal
wawa%
```

System tak odpowie jeśli nie czeka na Ciebie poczta. Jasne, że zamiast `jkowal` będzie *twójUID*.

Zwróć uwagę na dużą literę M. To nie wynika z mojego szacunku dla poczty. Niestety `Mail` znaczy co innego niż `mail`. Prawdę mówiąc w UNIX BSD nie są to różnice zauważalne od razu, uwierz, że są. Nie pytaj mnie dlaczego tak jest. Zagłębię się w dygresje i zgubię wątek.

Jeśli czeka na Ciebie poczta, to odpowiedź komputera może wyglądać tak:

```
wawa% Mail
Mail version SMI 4.1  Mon Sep 23 07:17 PDT 1991  Type ? for help.
"/usr/spool/mail/jkowal": 2 messages 2 new
>N  1 root          Mon Mar 28 13:41   15/405
 N  2 jsupel        Mon Mar 28 13:44   15/371   Przyjemnej pracy
&
```

Jest poczta! Nawet dwie. Program `mail` od razu podaje też sporo wiadomości. Najpierw przedstawia się wersją (czasami warto wiedzieć czy jest to `Mail` czy też `mailx`), uprzejmie informuje, że możesz wyświetlić na ekranie szybką pomoc. Informuje także, że w katalogu `/usr/spool/mail/jkowal` znajdują się dwie wiadomości, obie nowe czyli jeszcze nieczytane.

Dalej podaje dokładną listę wiadomości, jeden wiersz na jedną wiadomość, a w każdym wierszu listy:

- znak aktywności > (tylko przy jednej pozycji na liście),

spis listów

- stan (tutaj N-nowa),
- liczba kolejna na liście (bardzo ważny numerek, bo później, zamiast wypisywać całą nazwę, podaje się tylko ten numer),
- adres pocztowy nadawcy (tu wygląda dość prosto, bo poczta przesyłana była wewnątrz jednego komputera, bez używania sieci),
- data odbioru poczty,
- następne liczby oznaczają rozmiar otrzymanej wiadomości podany jako liczba linii/liczba znaków,
- na koniec jest informacja czego poczta dotyczy (to oczywiście wtedy, gdy nadawca wyszczególni [Subject:])

Całość informacji zakończona jest znaczkim **&** (ampersand). Jest to znak zachęty do wprowadzenia komendy (ze zbioru komend `mail`) i zlecenia wykonania następnego zadania dla programu poczty. Tym samym komendy właściwe dla powłoki C (`csh`) nie będą przyjmowane. Jest to najczęściej spotykany błąd początkujących, że próbują zlecić do wykonania np `ls`. Program oczywiście broni się przed wykonaniem polecenia, którego nie potrafi zinterpretować i wykonać, wysyła komunikaty o błędzie na ekran. Trzeba zwracać uwagę na to jaki jest znak zachęty[1.5] [prompt].

treść listu

Jak teraz zapoznać się z zawartością przychodzącej poczty?

Wystarczy po znaku **&** zachęty podać numer interesującego nas listu i nacisnąć klawisz *Enter* a na ekranie wyświetlona zostanie treść listu poprzedzona nagłówkiem (czasami dłuższym od samego listu):

```
& 1

Message  1:
From daemon Mon Mar 28 13:41:02 1994
Return-Path: <root>
Received: by wawa.noname (4.1/SMI-4.1)
id AA00206; Mon, 28 Mar 94 13:38:58 +0200
Date: Mon, 28 Mar 94 13:38:58 +0200
From: root (Operator)
Message-Id: <9403281138.AA00206@wawa.noname>
Apparently-To: jkowal
Status: R

Hej!
Zalozylem Panu konto uzytkownika "jkowal".
Panskie zbiory beda w katalogu /home/wawa/janek
Powodzenia w pracy.
Administrator systemu.

&
```

Proste? I łatwe.

Być może, po przeczytaniu tak miłego listu, chcielibyśmy natychmiast zaprosić BWO (czyli administratora systemu) do bufetu na kawę. Tylko jak zamknąć program **mail** bez strat? Najlepiej przez wydanie polecenia **x**, po prostu: *wyjście z poczty*

```
& x
wawa%
```

komputer wróci do powłoki i będzie czekał na rozkazy (już niekoniecznie pocztowe). Zapuśćmy pocztę jeszcze raz:

```
wawa% mail
Mail version SMI 4.1  Mon Sep 23 07:17 PDT 1991  Type ? for help.
"/usr/spool/mail/jkowal": 2 messages 2 new
>N  1 root            Mon Mar 28 13:41   15/405
 N  2 jsupel          Mon Mar 28 13:44   15/371   Przyjemnej pracy
&
```

Zawartość skrzynki pocztowej **/usr/spool/mail/jkowal nie uległa zmianie.** Nawet fakt przeczytania listu Nr 1 nie został odnotowany. **Oba listy są w stanie N.** Przeczytajmy drugi list:

```
& 2
Message  2:
From jsupel Mon Mar 28 13:44:05 1994
Return-Path: <jsupel>
Received: by wawa.noname (4.1/SMI-4.1)
id AA00220; Mon, 28 Mar 94 13:44:04 +0200
Date: Mon, 28 Mar 94 13:44:04 +0200
From: jsupel (Jerzy Supel)
Message-Id: <9403281144.AA00220@wawa.noname>
To: jkowal
Subject: Przyjemnej pracy
Status: R

Dzien dobry,
witam wsrod uzytkownikow systemu SUN IPPT.
Zycze przyjemnej pracy.
JS

&
```

polskie litery?

Pomijając brak polskich liter (niestety nie da się, później wytłumaczę się dlaczego) to też przyjemny list. Teraz już należałoby obu panów zaprosić na kawę. Przeczytaliśmy wszystkie listy, więc możemy zamknąć pocztę. Jeśli zamykamy pocztę poleceniem q to są tego konsekwencje:

```
& q
Saved 1 message in /home/wawa/janek/mbox
Held 1 message in /usr/spool/mail/jkowal
wawa%
```

wyjście q

Program `mail` zamknięty poleceniem q zostawił nieprzeczytaną wiadomość w pliku `/usr/spool/mail/jkowal` a tę przeczytaną schował (dołączył) do pliku `mbox`, który znajduje się w Twoim *home* i jeszcze wypisał komunikat informujący o tym.

Pamiętasz pewnie, że na starcie nie było żadych zbiorów w Twoim *home*. A teraz?

```
wawa% ls
mbox
```

mbox

Teraz pojawił się plik `mbox`, który zawiera Twoje przeczytane listy. Do tego pliku dołączane są sukcesywnie listy, które przeczytasz i zamkniesz program poleceniem q (chyba, że zdecydujesz co innego zrobić z przeczytanymi listami, ale o tym później[5.2.3]).

Sprawdźmy co jest w pliku `mbox`:

```
wawa% mail
Mail version SMI 4.1  Mon Sep 23 07:17 PDT 1991  Type ? for help.
"/usr/spool/mail/jkowal": 1 message 1 unread
>U  1 root                Mon Mar 28 13:41   16/415
&
```

W skrzynce pocztowej zarządzanej przez system (czyli w pliku systemowym o nazwie `/usr/spool/mail/jkowal`) został tylko komunikat (list) od administratora systemu (root), zaznaczony jako `U` (unread), czyli już nie nowy, bo był wywoływany na ekran, ale jeszcze nie czytany. Nie jest to do końca prawdą, bo przecież go czytalimy, ale po tamtym czytaniu listu od root zakończyliśmy pracę z programem `Mail` przez polecenie `x`, które zamyka program nie dokonując żadnych zmian.

Jeszcze jedna uwaga praktyczna.
Zdarza się, że list jest długi i "nie mieści się na ekranie". Wystarczy przypomnieć sobie o polecuniu more opisanym wcześniej[3.1] aby pozbyć się kłopotów. Wyjątkowo polecenie powłoki (uruchamiane po znaku przyzwolenia `%` lub `$`) może być wydane po znaku przyzwolenia poczty (`&`):

```
& more 1
...
```

spowoduje wyświetlenie na ekranie części listu nr.1. Przyciśnięcie klawisza *Enter* spowoduje wyświetlenie następnej porcji mieszczącej się na ekranie i tak do końca listu 1.

Ale tak naprawdę to trzeba właściwie skonfigurować zbiór `.mailrc` opisany w (5.6), wtedy nie będzie kłopotu z "uciekaniem ekranu". *.mailrc*

Cały czas program zachęca do skorzystania z pomocy (`Type ? for help`). Jak już pisałem wcześniej istnieje wiele programów pocztowych i dobrze jest skorzystać z pomocy, choćby dla sprawdzenia czy to jest to. Zobaczmy co daje użycie ? (nie zapomnij, że pomoc można wywołać z wewnątrz programu):

```
& ?

cd [directory] chdir to directory or home if none given
d [message list] delete messages
e [message list] edit messages
f [message list] show from lines of messages
h print out active message headers
m [user list] mail to specific users
n goto and type next message
p [message list] print messages
pre [message list] make messages go back to system mailbox
q quit, saving unresolved messages in mbox
r [message list] reply to sender (only) of messages
R [message list] reply to sender and all recipients of messages
s [message list] file append messages to file
t [message list] type messages (same as print)
top [message list] show top lines of messages
u [message list] undelete messages
v [message list] edit messages with display editor
w [message list] file append messages to file, without from line
x quit, do not change system mailbox
z [-] display next [previous] page of headers
! shell escape

A [message list] consists of integers, ranges of same, or user names
separated by spaces.  If omitted, Mail uses the current message.
&
```

Cała lista poleceń, które można zlecać programowi `mail` i `Mail` jeśli mamy znak zachęty [prompt] `&` (ampersand) czyli program znajduje się w trybie przeglądania.

Wykaz listów [message list] to nic innego jak numery kolejne listów (bez nawiasów oczywiście) np. `1 5 8` oznacza trzy listy o tych właśnie numerach a `3-6` oznacza cztery listy o numerach od 3 do 6. Można by podawać zamiast numerów nazwy (adresy) nadawców ale mało kto z tego korzysta.

Wypróbujmy kilka z poleceń. Dla skopiowania listu do pliku (innego niż `mbox`) użyjemy polecenia `s` :

```
& s 1 p1.s
"p1.s" [New file] 16/416
&
```

Program utworzył nowy plik o nazwie `p1.s` z zawartością listu No 1. Gdyby plik `p1.s` już istniał (i coś tam miał) to list No 1 będzie dołączony do końca istniejącej treści.

Podobnie zadziała polecenie `w` :

```
& w 1 p1.w
"p1.w" [New file] 14/378
&
```

Jest istotna różnica między działaniem tych dwu poleceń. `s` zostawia skrzynkę pocztową bez zmian, `w` powoduje zapisanie listu do nowego zbioru i usunięcie tego listu ze skrzynki.

Sprawdźmy jak wyglądają nasze (`jkowal`) skrzynki pocztowe:

```
& q
wawa% ls -l
total 10
-rw-------  1 jkowal        382 Mar 28 15:10 mbox
-rw-r--r--  1 jkowal        416 Mar 28 15:14 p1.s
-rw-r--r--  1 jkowal        378 Mar 28 15:15 p1.w
wawa% Mail
No mail for jkowal
wawa%
```

Wykonaliśmy dwa polecenia.

Pierwsze polecenie pokazuje jakie są aktualnie pliki w katalogu (zwróć uwagę na prawa dostępu do pliku `mbox`, może go czytać tylko właściciel konta, nawet `root` się nie dostanie). wszystkie te pliki są plikami tekstowymi, łatwymi do edycji i kopiowania.

Drugie przekonuje nas, że nie ma poczty dla `jkowal` (dokładnie plik systemowy `/usr/spool/mail/jkowal` jest chwilowo pusty) ale zapewniam Cię, że system dopilnuje tego aby nowa informacja przesłana dla `jkowal` została tam umieszczona.

Można natychmiast odpowiedzieć na list używając polecenia `r`, które zwalnia nas od wypisywania adresu nadawcy (na razie nie ćwiczyliśmy trybu pracy `mail` - wysyłanie poczty, o tym będzie dalej)

```
& r 1
To: root

Dziekuje za zyczenia
Jan Kowalski
```

```
EOT
&
```

Ale, aby objaśnić co wykonano, należy podać szczegóły pisania poczty.

5.2.2 Wysyłanie poczty

Wysyłanie poczty wiąże się z edycją tekstu. Można przygotować tekst niezależnym (od programu `Mail`) edytorem i użyć programu `Mail` tylko do wysyłki. Częściej jednak używa się `Mail` razem z wewnętrznym edytorem tekstu (domyślnie jest to `ex(1)`.

Najprościej jest uruchomić proces wysyłania poczty z poziomu powłoki (tzn. po znaku zachęty **%** ew. **$**):

```
wawa% Mail jkowal@wawa
... ...
```

gdzie w miejsce **jkowal@wawa** podać **adres użytkownika**[5.1] (lub grupę adresów), któremu chcesz przesłać list.

Możliwe jest też uruchomienie trybu wysyłania z wewnątrz trybu czytania (przeglądania) poczty (tzn. po znaku zachęty **&**) przez wydanie polecenia **m** *adres użytkownika*:

```
& m jkowal@wawa
... ...
```

Oba sposoby są równoprawne ale różnice zaznaczają się przy używaniu poleceń z tyldą[5.2.4] o czym później.

`Mail` uruchominy w trybie wysyłania zgłasza się pytaniem o temat listu [subject], który trzeba zawrzeć w jednej linii tekstu (możemy też pominąć). Po linii tematu wszystkie następne linie są treścią listu. Trzeba pamiętać, że używany edytor (`ex`) jest edytorem liniowym. To oznacza, że ew. poprawki są możliwe tylko w linii aktualnie pisanej. Każde naciśnięcie karetki (*Enter*) powoduje przejście do nowej linii i dopisanie poprzedniej linii do zbioru, bez możliwości jej poprawienia (chyba, że ... użyjesz poleceń z tyldą[5.2.4]). *poprawki*

Na ekranie zostaje oczywiście obraz linii ale edytor nie jest edytorem pełnoekranowym i nie można poruszać się kursorem po ekranie. Sposób poruszania się kursorem w obrębie jednej linii zależy od lokalnego komputera (terminala) i trudno tu opisać wszystkie możliwości. Po prostu trzeba spróbować. "Strzałki" najczęściej działają, gorzej z kasowaniem i wstawianiem.

Sygnałem do zakończenia edycji listu jest *Ctrl D* lub częściej używane - postawienie kropki (.) jako pierwszego (i jedynego) znaku w linii, zaraz po kropce ma być *Enter*. *koniec edycji*

Po zakończeniu listu program może zapytać o adresatów kopii (cc:) jeśli taką opcję wstawiliśmy do zbioru konfiguracyjnego `.mailrc`[5.6], podajemy adres(y)

w formacie takim samym jak dla odbiorcy (oddzielone spacją jeśli więcej niż jeden). Następnie program `Mail` przekazuje zakończony list do wysłania, robi jeszcze kilka innych rzeczy i przekazuje sterowanie do powłoki, czyli kończy pracę.

Kopia listu

Więcej szczegółów na temat wysyłania poczty (zwłaszcza opcje wywołania) możesz uzyskać korzystając z pomocy `man Mail`.

Zwrócę uwagę tylko na jedną opcję, która wydaje mi się bardzo użyteczna. Wywołanie poczty z opcją `-F` pozwala w łatwy sposób zachować kopię wysyłanego listu w oddzielnym zbiorze:

```
wawa% Mail -F jkowal@wawa
```

spowoduje (po zakończeniu procesu wysyłania) dołączenie treści listu do zbioru o nazwie `jkowal@wawa` jeśli taki zbiór wcześniej istniał, albo system otworzy nowy zbiór, nada mu nazwę `jkowal@wawa` i wpisze treść listu.

Oczywiście kopię listu można także otrzymać dodając swój adres do listy adresatów albo (częściej stosowane) adresując do siebie kopię `cc:` [carbon copy]. W obu przypadkach kopia listu zachowa się w zbiorze `mbox` ale w pierwszym przypadku adresowana do nadawcy w drugim do odbiorcy.

automatyczna kopia listu

Najlepszym sposobem zachowania kopii, w sposób automatyczny, jest odpowiednie ustawienie opcji w zbiorze `.mailrc`[5.6]. Wstawienie do tego zbioru linii

```
set record = ~/listy.wysłane
```

spowoduje kopiowanie (dołączanie) wychodzących listów do zbioru `listy.wysłane` w katalogu głównym.

Jeśli utworzymy folder pocztowy[5.2.3] to do zbioru `.mailrc` trzeba dodać linię zawierającą

```
set outfolder
```

a linię z `set record ...` zamienić na

```
set record = listy.wysłane
```

wtedy każdy wychodzący list będzie automatycznie dołączony do zbioru `listy.wysłane` w katalogu określonym przez `folder`.

5.2.3 Folder

Można przechowywać pocztę w zbiorach pocztowych nazywanych *folder*. Folderami są specjalne zbiory trzymane w *katalogu folder*. Dostęp do tych zbiorów jest możliwy bezpośrednio z programu `mail`. Foldery umożliwiają porządkowanie poczty przez podział pliku `mbox` na kilka plików, nie tracąc charakteru pliku pocztowego.

Dla skorzystania z folderu najpierw trzeba otworzyć katalog gdzie będą one przechowywane. Sprawdźmy najpierw jaka jest zawartość głównego katalogu:

```
wawa% cd
~/home/wawa/jsupel
wawa% ls
mbox test.txt
```

a następnie załóżmy nowy katalog (podkatalog katalogu głównego) o dowolnej, odpowiadającej nam nazwie:

```
wawa% mkdir poczta
wawa% ls -l
total 4
-rw------- 1 jsupel 944 Mar 31 15:34 mbox
drwxr-sr-x 2 jsupel 512 Apr 18 13:19 poczta
-rw-r--r-- 1 jsupel 54 Mar 23 11:08 test.txt
```

sprawdziliśmy od razu prawa dostępu do tego katalogu.

Następnym krokiem jest ustawienie ścieżki dostępu do katalogu poczta przez ustawienie zmiennej systemowej:

```
wawa% set folder=poczta
```

ustawienie folderu

Takie ustawienie będzie aktualne do czasu zamknięcia sesji roboczej. Aby uniknąć ciągłego ustawiania *set folder* należy wpisać tę zmienną do zbioru `.mailrc`[5.6] aby była wywoływana przy każdym otwarciu poczty.

Teraz już można posługiwać się folderem. Pokażę na przykładzie jak można utworzyć nowe *foldery*. Najpierw sprawdźmy co jest w skrzynce pocztowej:

```
wawa% Mail
Mail version SMI 4.1-OWV3 Mon Sep 23 07:17:24 PDT 1991
Type ?  for help.
"/usr/spool/mail/jsupel":  4 messages 2 new
1 witold Wed Mar 23 00:18 15/341 inny test
2 To witold Wed Mar 23 00:40 17/365 jeszce raz test
>N 3 jkowal Mon Apr 18 12:59 13/317 proba folderu poczta
N 4 witold Mon Apr 18 13:03 26/621 tetst attach
&
```

a następnie (będąc w dalszym ciągu w trybie czytania poczty) schowajmy trzecią informację do folderu o nazwie janek. Znak plus (+) przed nazwą folderu informuje program, że właśnie chodzi o folder:

zapisanie do folderu

```
& s 3 +janek
"/home/wawa/jsupel/poczta/janek" [New file] 13/327
```

odpowiedzią programu jest informacja o przeniesieniu listu nr.3 do zbioru o nazwie janek w katalogu /poczta, a ponieważ takiego zbioru nie było, program otworzył nowy zbiór o takiej nazwie. Gdyby zbiór janek już istniał to list nr.3 byłby dopisany do isniejącej zawartości, na końcu zbioru.

Inny list możemy dołączyć do innego folderu:

```
& s 4 +witek

"/home/wawa/jsupel/poczta/witek" [New file] 26/631 &
```

Podobnie jak poprzednio utworzony został nowy zbiór (a właściwie folder bo dodaliśmy plus (+) przed nazwą)

Jakie są dostępne foldery pocztowe można dowiedzieć się wydając polecenie **folders**:

```
& folders

janek witek
```

które wylistuje wszystkie foldery w katalogu **poczta** - aktualnie wskazanym jako katalog z folderami pocztowymi.

Zakończenie sesji z programem **Mail** spowoduje zachowanie zmian:

```
& q

Held 2 messages in /usr/spool/mail/jsupel
```

wyjście z programu przez **x** zostawiłoby wszystkie cztery listy w skrzynce listów przychodzących [in-box] (**/usr/spool/mail/jsupel**) jednocześnie zachowując kopie informacji w folderach, gdzie zostały wcześniej przesłane.

Sprawdźmy zawartość katalogu **poczta**:

```
wawa% cd poczta

/home/wawa/jsupel/poczta
wawa% ls -l
total 2
-rw-r--r-- 1 jsupel 327 Apr 18 13:20 janek
-rw-r--r-- 1 jsupel 631 Apr 18 13:20 witek
```

Są więc dwa zbiory, których charakter jest taki sam jak zbioru **mbox** w katalogu głównym (zawierają treść listów z nagłówkami właściwymi dla zbiorów pocztowych). Ich właściwością jest to, że będą zawierać tylko te listy, które tam skierujemy, podczas gdy **mbox** zawiera listy "jak leci" dołączane automatycznie po zakończeniu programu instrukcją **q**.

przeglądanie folderu

Pliki-foldery są zwykłymi plikami tekstowymi i można je "obrabiać" dowolnym edytorem tekstowym. W szczególności jednak mogą być wywoływane programem pocztowym:

```
wawa% Mail -f +witek
Mail version SMI 4.1-OWV3 Mon Sep 23 07:17:24 PDT 1991
Type ? for help.
"+witek": 1 message 1 new
> 1 witold Mon Apr 18 13:03 27/631 tetst attach
&
```

co umożliwia dalszą pracę w trybie czytania poczty.

Powrót do aktualnej skrzynki pocztowej [in-box] możliwy jest bez wychodzenia z programu Mail:

```
& file %
"/usr/spool/mail/jsupel": 2 messages
&
```

Powrót do pierwotnego folderu odbywa się po poleceniu:

```
& file #
"+witek": 1 message 1 new
&
```

Jeśli to wszystko wydaje się skomplikowane to dowiedz się, że będąc w trybie czytania poczty (znak zachęty &) można zawsze wylistować dostępne foldery: *dostępne foldery*

```
& folders
janek witek
&
```

i przejść do dowolnie wybranego folderu:

```
& folder +witek
"+witek": 1 message
&
```

(nie zapomnij o plusie przed nazwą folderu).

Zalety pracy z folderami docenisz przy większej liczbie otrzymywanych i wysyłanych listów, gdy zachodzi konieczność sortowania wg. respondentów lub tematów.

5.2.4 Polecenia z tyldą

W trybie wysyłania poczty, gdy edytujemy list do wysłania, dostępnych jest kilka użytecznych poleceń "z tyldą". Jeśli pierwszym znakiem w linii jest znak tyldy (~) to należy oczekiwać wykonania polecenia (jakie to polecenie, decydują znaki następujące po znaku tyldy), którego skutki nie zapisują się w buforze edytowanego listu (chyba, że ma być inaczej) i nie są umieszczane w wysyłanym liście. W szczególności może to być znak zapytania, po którym wypisywane są komendy "z tyldą", z krótkim objaśnieniem. Spróbujmy wysłać list i użyć komendy z tyldą:

```
wawa% Mail jkowal@wawa
Subject:test
tutaj treść listu
?  to co poniżej pojawia się na ekranie nie wchodzi do listu!
-------------------- ~ ESCAPES ----------------------------
~~            Quote a single tilde
~a,~A         Autograph (insert 'sign' variable)
~b users      Add users to Bcc list
~c users      Add users to Cc list
~d            Read in dead.letter file
~e            Edit the message buffer
~m messages   Read in messages, right-shifted by a tab
~f messages   Read in messages, do not right-shift
~h            Prompt for To list, Subject and Cc list
~p            Print the message buffer
~q,~Q         Quit, save letter in $HOME/dead.letter
~x            Quit, do not save letter
~r file       Read a file into the message buffer
~s subject    Set subject
~t users      Add users to To list
~v            Invoke display editor on message
~w file       Write message onto file
~.            End of input
~?            Print this message
~!command     Run a shell command
~|command     Pipe the message through the command
~:command     Execute regular Mail command
-------------------------------------------------------------
```

Po wyświetleniu spisu komend program znów wraca do trybu wczytywania (przyjmowania znaków z klawiatury). Możemy dla przykładu posłużyć się komendą ~r filename, *która powoduje przyłączenie wskazanego zbioru do pisanego listu:*

```
~r test.txt
"test.txt" 4/54

EOT
wawa%
```

O dołączeniu zbioru `test.txt` do listu program poinformował nas jedną linijką, bez wyświetlania na ekranie zawartości dołączanego zbioru. Po wstawieniu kropki w pustej linii zakończyliśmy edycję i spowodowaliśmy wysłanie redagowanego listu.

Inne komendy z tyldą proponuję wypróbować we własnym zakresie.

5.3 mailx w UNIX VR4

Działanie programu **mailx**, który jest, w systemie Solaris 2.x (UNIX VR4), podstawowym programem obsługi poczty dla użytkownika, nie różni się w sposób istotny od działania wcześniej opisanego programu Mail. Wywołanie polecenia Mail, w maszynach pracujących pod systemem VR4, powoduje uruchomienie programu **mailx**. Tak więc uruchomienie poczty przez Mail [*opcje*] lub **mailx** [*opcje*] jest równoważne.

Pewne różnice są przy rozkazach "z tyldą" ale wynikają jedynie z rozszerzenia w **mailx** liczby komend. Można to sprawdzić przez wpisanie ne początku linii polecenia ~? gdy jesteśmy w trakcie redagowania listu.

5.4 [aliases]

Pocztowy **alias** jest grupą wybranych adresów pocztowych, której została nadana jedna (wspólna) nazwa.

Użyj mechanizmu **alias** jeśli często wysyłasz pocztę do tej samej grupy użytkowników. Jeśli, dla przykładu, chcesz wysyłać zawiadomienia o seminariach pracownianych do ludzi, których adresami są: **jkoval@wawa, witold, rt@ims.ifta.edu.be** to utwórz alias o nazwie, powiedzmy, **sem_prac**. Za każdym razem gdy wyślesz pocztę (np. zawiadomienie) pod adres **sem_prac** automatycznie wyślesz ją do trzech ludzi. *etc/alias*

Są dwie różne drogi do stworzenia **aliasów**. Jedne mogą być zapisane w Twoim zbiorze **.mailrc**[5.6], drugie w zbiorze systemowym **/etc/aliases**. Niestety użycie tych dwu rodzajów różni się nieco. Zacznijmy od omówienia tych lokalnych.

Zbiór **.mailrc** możesz znaleźć w Twoim głównym katalogu. Zawiera pewną liczbę opcji określających styl pracy z **Mail**, więcej o tym zbiorze w p.(5.6). *alias*

Jeśli już masz ten zbiór **.mailrc** (pamiętaj, że dla sprawdzenia obecności tego zbioru trzeba dać polecenie **ls -a**) to możesz dodać do niego grupę adresów (np. tę dotyczącą seminarium) i jej **alias**. Używając dowolnego edytora dodaj do zbioru **.mailrc** (najlepiej na końcu zbioru) następującą linię: *.mailrc*

```
alias sem_prac jkoval@wawa witold rt@ims.ifta.edu.be
```

opis aliasu zawiera więc: a) słowo alias b) nazwę grupy, będącą jej adresem pocztowym c) adresy pocztowe uczestników grupy oddzielone spacjami. Jeśli wykaz uczestników jest długi i nie mieści się na ekranie – nie przejmuj się – kontynuuj pisanie w linii, aż do wyczerpania listy (alias w **.cshrc** nie może zawierać, między adresami, innych znaków niż spacja, nawet niewidocznych).

- Przy wysyłaniu poczty podasz jako adres tylko **alias** (nazwę grupy) bez dodawania nazwy maszyny (np. tylko **sem_prac**).

- Aliasy w **.mailrc** są prywatne. To znaczy, że tylko Ty możesz ich użyć. Ktoś inny (z innego konta) wysyłając pocztę na adres **sem.prac** dostanie odpowiedź "user unknown".

- Aliasy `.mailrc` są automatycznie rozwijane. Każdy z uczestników grupy otrzyma list w takiej postaci jak by był on wysyłany indywidualnie. Fakt użycia **alias** nie jest odnotowywany.

Poniżej różnice aliasów lokalnych (`.mailrc`) i ogólnych (`/etc/aliases`):

- Aliasy wykazane w zbiorze `/etc/aliases` są własnością publiczną. Każdy (nawet z innej sieci LAN) może wysłać pocztę na alias z tego zbioru.
- Możesz zobaczyć jakie są dostępne aliasy listując zbiór `/etc/aliases` ale nie możesz ich zmieniać ani uzupełniać (bez uprawnień superużytkownika).
- Zauważysz nieco inny sposób zapisu w zbiorze **aliases**: brak słowa alias, po nazwie grupy jest średnik, adresy oddzielone są przecinkami i mogą znajdować się w wielu liniach.
- Wysyłając pocztę na adres ze zbioru **aliases** powinieneś dodać nazwę komputera, który przetrzymuje ten zbiór, np. jeśli na maszynie **lks** w zbiorze `/etc/aliases` jest alias *zmcig* to adres będzie **zmcig@lks**.
- Indywidualny adresat jest poinformowany, że list trafił do niego przez **alias**.

5.5 Inne sprawy

Jest kilka programów związanych z pocztą (*E-mail*) ale uruchamianych niezależnie od programu **mail**.

skąd poczta?

Jeśli nie chcesz sprawdzać swojej poczty przez `Mail`, możesz użyć polecenia from, które wyświetli na ekranie zawartość skrzynki pocztowej, np.:

```
wawa% from
From witold@spoko Mon Apr 18 13:03:15
.....
wawa%
```

i wróci do powłoki (można dalej wydawać polecenia systemowi). To samo (a nawet więcej bo z rozmiarem listu i opisem zawartości) osiąga się uruchamiając program **Mail** bez opcji a następnie wpisując **x** po wyświetleniu nagłówków poczty. Pewnie do czegoś to było potrzebne, teraz **from** jest mało używane. Wspominam dla porządku. Na wszelki wypadek powiem, że więcej dowiesz się jak zwykle po **man** from.

Innym udogodnieniem, bardzo użytecznym, jest program **vacation**. Poczta przychodzi do Ciebie nawet podczas dłuższej nieobecności. Nawet wyłączenie komputera też nie zatrzymuje działania poczty. W sumie jest to bardzo wygodne. Po powrocie możesz przejrzeć odłożoną pocztę. Są jednak przypadki gdy istotne jest aby nadawca wiedział, że poczta nie została przeczytana. Wprawny nadawca, jeśli mu na tym zależy, może sprawdzić czy czytasz

pocztę. Użyje do tego polecenia finger, które między innymi informuje kiedy poczta była czytana[4.4]. Inne sposoby, właściwe administratorom poczty, pomijam.

co zrobić z pocztą gdy jesteśmy dłużej nieobecni

Można się zabezpieczyć przed niepewnymi sytuacjami i włączyć coś w rodzaju automatycznej sekretarki, która będzie informowała nadawcę, że aczkolwiek poczta doszła do Ciebie to przeczytasz ją później. Program vacation wyśle do nadawcy, uprzednio przygotowaną, informację przechowywaną w zbiorze tekstowym `.vacation.msg` (zbiór ten znajdziesz w swoim *home*). Możesz ten zbiór dowolnie zmieniać, dowolnym edytorem, z jednym zastrzeżeniem: pierwsza linia musi zaczynać się od `Subject:`. Poleceniem `vacation` uruchamiasz interaktywny program, który poprowadzi Cię za rączkę:

```
wawa% vacation
This program can be used to answer your mail
automatically when you go away on vacation.
You need to create a message file in
/home/wawa/jsupel/.vacation.msg first.  Please use your
editor (/usr/ucb/vi) to edit this file.
...
```

Dalej trzeba odpowiadać na pytania. Pewnym utrudnieniem może być użycie edytora `vi`[8.0] do stworzenia zbioru `.vacation.msg`. Jeśli ograniczysz się do standardowo zaproponowanego tekstu, to po automatycznym włączeniu edytora wykonaj sekwencję: `Escape : wq` co spowoduje zapisanie zbioru w proponowanej postaci.

Program `vacation` tworzy automatycznie, w katalogu *home*, jednolinijkowy zbiór `.forward`, o zawartości w moim przypadku:

```
\jsupel, "|/usr/ucb/vacation jsupel"
```

którego obecność jest warunkiem działania automatycznych odpowiedzi.

Powrót z urlopu i powrót do normalnego stanu poczty wymaga ponownego uruchomienoa programu `vacation`, który jest na tyle inteligentny (inteligencja to jest to! w UNUXie), że domyśli się iż chodzi o likwidację automatu.

Oczywiście więcej o `vacation` można się dowiedzieć z `man vacation`.

Przy okazji, zbiorek `.forward` można wykorzystać, niezależnie od programu `vacation`, do przesyłania kopii Twoich listów do innego użytkownika (a nawet do innego komputera). Jeśli zbiór `.forward` będzie wyglądał np. tak:

automatyczne przeadresowywanie poczty

```
\jkowal, /home/janek/schowek.rezerwowy.poczty
```

to cała poczta przychodząca do Janka Kowalskiego będzie kopiowana do zbioru `schowek.rezerwowy.poczty` niezależnie od kopii listów trzymanych w `mbox` i `/usr/spool/mail/jkowal`. Pamiętaj, że plik `schowek...` (pieruńsko długą nazwę wymyśliłem) musi już istnieć i być dostępny do pisania dla wszystkich[3.4] a więc zrób:

```
wawa% touch schowek.rezerwowy.poczty
wawa%
```

co utworzy pusty plik[3.1] o tej nazwie (założyłem, że jesteś w *home*) a następnie zmień prawa dostępu:

```
wawa% chmod a+w schowek.rezerwowy.poczty
wawa%
```

5.6 Plik .mailrc

Ten plik zawiera ustawienie kilku parametrów, które ustalają styl pracy z `mail` (i `MailTool` jeśli także pracujesz w oknach). Zmieniając ustawienie parametrów (plik jest plikiem tekstowym) można dostosować własności programu `mail` do własnych potrzeb.

Najpierw sprawdź czy masz ten program w swoim głównym katalogu *home* (np. `more .cshrc` lub `ls .cshrc`). Jeśli nie, możesz skopiować wzorcowy:

```
wawa% cp /usr/lib/Mailrc .mailrc
wawa%
```

i staniesz się właścicielem zbioru, który przytoczę w całości:

```
set alwaysignore
set askcc
set asksub
set autoprint
set cmd="lpr -p &"
set crt=15
set DEAD=~/dead.letter
set EDITOR=/usr/ucb/ex
set hold
set indentprefix="> "
set keepsave
set metoo
set PAGER="cat -s | more -22 -c"
set prompt="{Mail}& "
set record=~/.record
set SHELL=/bin/csh
set VISUAL=/usr/ucb/vi

ignore apparently-to date errors-to from id in-reply-to \
       message-id precedence received references remailed-date \
       remailed-from return-path sent-by status via
```

Poszczególne linie w tym zbiorze oznaczają:

alwaysignore powoduje ograniczenie informacji w nagłówku, niezależnie od ustawienia parametrów **ignore**, przy kopiowaniu poczty do zbioru.

askcc po zakończeniu pisania listu automatycznie pyta "Cc:" [carbon copy] o adres(y) przesłania kopii. Domyślne ustawienie [off].

asksub przed pisaniem treści listu zapyta czego dotyczy (subject:). Domyślne ustawienie [on].

autoprint samoczynnie wyświetla następny list jeśli poprzedni skasowałeś. Domyślnie [off].

cmd="lpr -p &" domyślna komenda powłoki dla komendy **pipe** wewnątrz **Mail**. Raczej dla zaawansowanych.

crt=15 liczba linii tekstu poczty wyświetlanych na ekranie. Jeśli list jest dłuższy to strumień jest kierowany na ekran zależnie od polecenia określonego przez PAGER (zwykle użyte jest **more**). Domyślnie wyłączone ([off]) i dlatego długi list "gubi" początek. Działa tylko w **Mail**.

DEAD=~/dead.letter to jest pełna nazwa zbioru, w którym chowany jest list w przypadku awaryjnego przerwania pisania.

EDITOR=/usr/ucb/ex nazwa edytora używanego przy redagowaniu poczty.

hold jeśli włączone to przeczytane listy nie znikają (nie są automatycznie przesuwane do zbioru **mbox**) ze skrzynki natychmiast po ich skopiowaniu do innego zbioru lub skasowaniu. Domyślne ustawienie jest różne w **mail** i **Mail**, stąd wrażenie, że programy te działają w różny sposób.

indentprefix="> " pisząc list możemy dołączyć inny list. Dla odróżnienia go od listu pierwotnego będzie on, w każdej linii, poprzedzony znakami, które tu są zdefiniowane między "pazurkami".

keepsave normalnie list kopiowany do zbioru lub do folderu jest kasowany z aktualnej skrzynki pocztowej, **keepsave** zachowuje list.

metoo jeśli wysyłasz list do grupy użytkowników używając opcji [alias] i sam jesteś jesteś jej członkiem, to nie otrzymasz kopii jeśli nie masz **metoo** ustawionego [off]. To działa tylko dla grup **alias** zadeklarowanych w własnym **.mailrc**. Jak zadziała dla grup systemowych wie tylko BWO (Bardzo Ważna Osoba) czyli Pan Administrator Systemu.

PAGER="cat -s | more -22 -c" tej komendy **mail** używa do wyświetlania długich listów (patrz wyżej opcja **crt**)

prompt="{Mail}& " zwyczajowo znak zachęty ogranicza się do **&**, między "pazury" możesz wpisać co chcesz.

`record=~/.record` nazwa zbioru do przechowywania kopii każdego wysłanego listu. Jeśli jest otwarty `outfolder` to zbiór `record` będzie ulokowany w folderze. Domyślnie [off].

`SHELL=/bin/csh` nazwa preferowanego interpretera poleceń (powłoki). Domyślnie `sh`.

`VISUAL=/usr/ucb/vi` w tym przypadku będzie użyty edytor `vi` jeśli użyjesz komendy `~v` (i niech Cię Pan Bóg ma w opiece).

`ignore` powoduje skrócenie nagłówka poczty o wymienione opcje.

Jest jeszcze kilkadziesiąt innych opcji, które można ustawić w `.mailrc`. Nie warto o wszystkich tu pisać, bo albo używa się prostego `Mail` dla jego prostoty, albo używa się specjalnych programów pocztowych z wodotryskiem itp. albo czyta się dokumentację źródłową.

Niemniej jednak o kilku jeszcze opcjach powinienem powiedzieć.

`folder` wstawienie do `.mailrc` linii `set folder=~/listy` otworzy katalog dla przechowywania zbiorów pocztowych ([folder]).

`outfolder` wstawienie tej opcji pozwala na automatyczne chowanie wysyłanego listu do zbioru określonego opcją `record` (ale przy włączonym `outfolder`).

`expert` taką opcję ustawisz gdy poczujesz się wprawnym użytkownikiem poczty i nie będziesz zawracać sobie głowy jakimiś głupimi ostrzeżeniami systemu. Domyślnie [off]

Pamiętaj, że to co powiedziałem o opcjach ustawianych w zbiorze `.mailrc` dotyczy poczty używanej z programem `mail` lub `Mail` albo `mailx`. Poczta w oknach `MailTool` może zachowywać się nieco inaczej (pomijając różnice wynikające z grafiki). W szczególności jednak opcje związane z `folder` i `record` (automatyczne zapisywanie kopii wysylanego listu) będą działać także w oknach.

Rozdział 6

FTP (wymiana plików)

FTP [File Transfer Protocol] jest podstawowym sposobem przesyłania zbiorów między komputerami sieci Internet. W wielu systemach operacyjnych jet to także nazwa programu obsługującego protokół `ftp`. Mając odpowiednie zezwolenia można skopiować zbiór z komputera w Afryce Południowej do komputera w Stanach Zjednoczonych siedząc w dodatku przed komputerem w Polsce. Szybkość transmisji jest w granicach 5-10 K/sek, zależy jednak od szybkości łączy po drodze i natężenia ruchu. W warunkach polskich, gdy podstawowym łączem jest modem 9600 bps, prędkość transmisji zza oceanu, w okolicy południa, nie przekracza 200 bajtów/sek. *9600 bps*

FTP jednak jest bardzo silnym narzędziem, jego zalety dostrzegane są także w pracy sieci lokalnej (LAN) gdyż pozwala na transfer zbiorów między komputerami różnych systemów (w szczególności między DOSowym PCetem i UNIXową Work Station). Jak zwykle możemy przeczytać opis programu używając polecenia `man ftp` (w komputerach UNIXowych).

6.1 Uzyskanie połączenia

Połączenie, w protokole FTP, z innym komputerem można uzyskać dwoma drogami: przez wywołanie nazwy pożądanego komputera [hostname] lub adresu Internet. Preferowane jest użycie nazwy. Tak więc poprawną formą wywołania połączenia z drugim komputerem będzie:

```
wawa% ftp lks.ippt.gov.pl
```

choć równoważne jest

```
wawa% ftp 148.81.52.7
```

Po chwili (zależnej od odległości wywoływanego komputera jak i szybkości łączy po drodze) ukaże się komunikat:

```
Connected to lks.ippt.gov.pl
```

a następnie przedstawi się system operacyjny wywoływanego komputera, któty zapyta o nazwę użytkownika i hasło. W przypadku gdy przyłączamy się do

komputera na którym mamy swoje konto lub znamy jakieś konto i hasło, sprawa jest rutynową, trzeba odpowiedzieć na pytania.

Jeśli wywołujemy komputer z sieci lokalnej to nie jest konieczne podawanie pełnej nazwy (z domeną) wystarczyłoby tylko `ftp lks`.

Komputery PC-owe, jeśli dołączone są do sieci Internet i są wyposażone w program FTP w wersji DOS, też mogą łączyć się z dowolnym innym komputerem w sieci Internet. Wywołanie programu `ftp` odbywa się z poziomu systemu operacyjnego DOS:

```
C:\ >ftp lks.ippt.gov.pl
```

i dalej sesja przebiega tak samo jak dla innych systemów.

serwer FTP

Jeśli dołączamy się do *servera ftp* (są takie komputery, które udostępniają swoje zbiory do kopiowania) to zwyczajowo konto użytkownika (dostępne dla wszystkich) nazywa się **anonymous** a hasłem [Password] będzie Twój adres Internet. Czyli ja w tym miejscu piszę (oczywiście nie ma echa hasła na ekranie):
`Password:` **jsupel@ippt.gov.pl**
po prawidłowym wpisaniu nazwy użytkownika i hasła odpowiedzią jest znak zachęty:

```
ftp>
```

Od tego momentu polecenia wprowadzone po znaku zachęty `ftp>` są interpretowane w lokalnym komputerze i przesyłane do wykonania w zdalnym komputerze (jeśli dotyczą zdalnego komputera). W szczególności można skorzystać z wewnętrznej pomocy: **ftp>help**

Krótkie (jednowierszowe) informacje o tych poleceniach można uzyskać przez:

```
ftp> help polecenie
```

Poniżej wykaz komend programu `ftp` zaimplementowanego dla systemu SunOS 4.1:

```
!           escape to the shell
$           execute macro
account     send account command to remote server
append      append to a file
ascii       set ascii transfer type
bell        beep when command completed
binary      set binary transfer type
bye         terminate ftp session and exit
case        toggle mget upper/lower case id mapping
cd          change remote working directory
cdup        change remote working directory to parent directory
close       terminate ftp session
cr          toggle carriage return stripping on ascii gets
delete      delete remote file
debug       toggle/set debugging mode
```

```
dir         list contents of remote directory
disconnect terminate ftp session
form        set file transfer format
get         receive file
glob        toggle metacharacter expansion of local file names
hash        toggle printing '#' for each buffer transferred
help        print local help information
lcd         change local working directory
ls          nlist contents of remote directory
macdef      define a macro
mdelete     delete multiple files
mdir        list contents of multiple remote directories
mget        get multiple files
mkdir       make directory on the remote machine
mls         nlist contents of multiple remote directories
mode        set file transfer mode
mput        send multiple files
nmap        set templates for default file name mapping
ntrans      set translation table for default file name mapping
open        connect to remote tftp
prompt      force interactive prompting on multiple commands
proxy       issue command on alternate connection
sendport    toggle use of PORT cmd for each data connection
put         send one file
pwd         print working directory on remote machine
quit        terminate ftp session and exit
quote       send arbitrary ftp command
recv        receive file
remotehelp get help from remote server
rename      rename file
reset       clear queued command replies
rmdir       remove directory on the remote machine
runique     toggle store unique for local files
send        send one file
status      show current status
struct      set file transfer structure
sunique     toggle store unique on remote machine
tenex       set tenex file transfer type
trace       toggle packet tracing
type        set file transfer type
user        send new user information
verbose     toggle verbose mode
?           print local help information
```

6.2 Użyteczne polecenia FTP

Z tych 58 poleceń programu `ftp` wykorzystujemy praktycznie tylko kilka. Przede wszystkim:

```
ftp> quit
```

kończy pracę (i rozłącza komputery), to samo daje `bye`.

Polecenia `disconnect` i `close` odłączają obcy server ale pozostawiają wewnątrz programu `ftp`.

Do skopiowania obcego zbioru (z obcego komputera na swój) użyjemy polecenia `get` (lub jego synonimu `recv`):

```
ftp> get zdalny.plik [lokalny.plik]
```

pamiętając, że jeśli lokalną maszyną jest PCet to nazwa lokalnego pliku ma być właściwa dla DOSu.

zbiory binarne

Tak łatwo kopiuje się pliki tekstowe, nie zawierające kodu powyżej 127 (czyli np. polskich liter w kodzie Mazovii). Dla skopiowania pliku binarnego trzeba ustawić typ zbioru przez `binary`:

```
ftp> binary
ftp>
```

od tego momentu przekazywane (w obie strony) mogą być także mapy bitowe.

Przesłanie swojego zbioru na obcy komputer odbywa się poleceniem `put` (lub synonimem `send`):

```
ftp> put lokalny.plik [zdalny.plik]
```

W szczególności można przesłać zbiór z dyskietki PCta do servera UNIXowego wydając polecenie np.:

```
ftp> send a:\plik.txt plik.z.tekstem.poczty
200 PORT command successful
150 ACII data connection for plik.z.tekstem.poczty
(192.9.200.1,1157)
226 ASII Transfer complete
local: plik.txt remote:plik.z.tekstem.poczty
378 bytes sent in 0.043 seconds (8.6 Kbytes/s)
ftp>
```

Jak widać program bardzo szczegółowo informuje o sposobie wykonania polecenia.

Rozdział 7

Drukowanie

Drukownie nie jest sprawą prostą. Różnorodność sprzętu (drukarek) powoduje, że jest to problem indywidualny, zależny w dużym stopniu od własności drukarki.

W zamierzchłej przeszłości (u pczątków UNIXa, w latach 70-tych) drukowano na drukarkach wierszowych (olbrzymie, hałaśliwe, ciężkie, drogie, skomplikowane, szybkie, drukujące tylko litery, cyfry i trochę znaków specjalnych) obsługiwanych przez specjalizowany personel.

W tym czasie miały zastosowanie polecenia systemowe lpr lpq. Wydawało się polecenie:

```
% lpr plik.txt
```

i już. Zbiór tekstowy, (tylko takie dały się drukować) drukowany był na papierze "składance" i odkładany na stos razem z wydrukami innych. Nic więc dziwnego, że wydruk musiał być poprzedzony nagłówkiem [header] na całą stronę, z danymi drukującego i nazwą zbioru (tym zajmował się system).

Od chwili pojawienia się drukarek graficznych, najpierw trmicznych i igłowych (mozaikowych), później laserowych i atramentowych, drukowanie stało się przyjemnością ale i udręką równocześnie. Wraz z grafiką pojawiły się problemy standaryzacji formatu graficznego i języka komend (do czasu narzucenia standardu przez firmy Hewlett Packard (PCL) i Adobe (PostScript).

W UNIXie dalej używa się prostych poleceń typu lp do drukowania (ściślej przesyłania zbioru do drukarki). Sprawą Administratora Systemu jest odpowiednie przyłączenie drukarek w systemie. Patrz jak kwaśną minę ma człowiek odpowiedzialny za tę sprawę gdy go niewinnie zapytasz "jak drukować". Po tej minie szybko się zorientujesz jak się ma sprawa.

Tak w ogóle, to mogą być drukarki sieciowe (coś jak kiedyś wierszowe ale oczywiście to nie to samo) i drukarki lokalne. Te ostatnie będą przyłączone do komputera przy którym siedzisz lub gdzieś w pobliżu (jak widać pojęcie "lokalna" można brać dosłownie - w tym lokalu). Sprawdź w systemowym zbiorze `/etc/printcap` co jest Twoją drukarką lokalną. */etc/printcap*

Dla utrudnienia życia polecenia drukowania różnią się w zależności od powłoki [shell].

7.1 Drukarka w powłoce csh

Przypomnę, że powłoka[1.5] **csh** [c shell] to ten interpreter poleceń, który zgłasza się znakiem zachęty **%**. Gdyby były jakieś wątpliwości co do powłoki to można ją sprawdzić poleceniem **set** (bez argumentu).

Poniżej przykłady najmniej wymyślnych poleceń związanych z drukowaniem:

`lpr` *plik*	drukuje zbiór o nazwie *plik* na domyślnej drukarce, doda na początek stronę informacyjną,
`lpr -h` *plik*	jak wyżej ale bez strony informacyjnej,
`lpr -P`*drukarka plik*	drukuje zbiór o nazwie *plik* na drukarce o nazwie *drukarka* (zwróć uwagę na brak spacji między opcją `-P` a nazwą drukarki). Nazwę drukarki znajdziesz w zbiorze `/etc/printcap` lub dowiesz się u Administratora,
`lpr -hP`*drukarka plik*	to samo co wyżej ale bez dodatkowej strony informacyjnej [banner],
`lpq`	(bez argumentu) otrzymasz raport o aktualnie drukowanych i czekających w kolejce zadaniach w drukarce domyślnej,
`lpq -P`*drukarka*	raport dotyczy wskazanej drukarki,
`lprm` *job*	usuniesz z kolejki zadanie o numerze *job*, który znasz z raportu `lpq`,
`lprm --`	skasujesz wszystkie zadania drukowania.

Dla bardziej dociekliwych dodam, że znając nazwę drukarki np. *drukarka* umieszczonej w dowolnym miejscu sieci możemy ustawić zmienną systemową PRINTER:

drukarka domyślna

% setenv PRINTER *drukarka*

i na czas sesji roboczej *drukarka* staje się domyślną, czyli polecenie `lpr` *plik* wyśle wydruk na tę właśnie drukarkę.

7.2 Drukarka w powłoce sh

Przypomnę, że powłoka[1.5] **sh** to ten interpreter poleceń, który zgłasza się znakiem zachęty **$**.

Poniżej przykłady niektótych poleceń związanych z drukowaniem:

`lp` *plik*	drukuje zbiór o nazwie *plik* na domyślnej drukarce, doda na początek stronę informacyjną,
`lp -o nobanner` *plik*	jak wyżej ale bez strony informacyjnej,
`lp -d` *drukarka plik*	drukuje zbiór o nazwie *plik* na drukarce o nazwie *drukarka* (zwróć uwagę na obecność spacji między opcją `-d` a nazwą drukarki). Nazwę drukarki znajdziesz w zbiorze `/etc/printcap` lub dowiesz się u Administratora,
`lp -h` *plik*	doda nagłówek na każdej stronie,
`lp -n4` *plik*	wydrukuje cztery kopie,
`lp -w -d` *drukarka plik*	wypisze, po zakończonym drukowaniu, komunikat na ekranie (jeśli `-m` zamiast `-w` to komunikat wyśle *E-mailem*),
`lpstat`	(bez argumentu) otrzymasz raport o aktualnie drukowanych i czekających w kolejce zadaniach ,
`lpstat -s`	raport o dostępnych drukarkach,
`lpstat -`*opcja*	inne opcje znajdziesz przez man lpstat,
`cancel` *drukarka*	przerwiesz aktualne drukowanie,
`cancel` *requestid*	skasujesz wskazane zadanie drukowania.

Można zmienić domyślną drukarkę przez ustawienie zmiennej systemowej PRINTER: *drukarka domyślna*

```
$ PRINTER = drukarka
$ export PRINTER
```

od tej chwili drukarka o nazwie *drukarka* jest domyślną.

7.3 Inne sposoby drukowania

Może okazać się, że najprostszym sposobem wydrukowania zbioru UNIXowego będzie skopiowanie go na dyskietkę[4.8] lub skopiowanie na dysk komputera PC przy pomocy programu FTP[6.0] i wydrukowanie na drukarce PCtowej.

GhostScript

W przypadku zbiorów PostScriptowych można skorzystać z programu Ghost Script (jeśli taki jest udostępniony w systemie) i przygotować zbiór do druku na drukarce laserowej (typu HP) lub igłowej (typu Epson). Jeśli już mamy zbiór w standardzie PostScript np. o nazwie `mój.ps` to przetłumaczymy go na format zrozumiały dla zwykłej drukarki w sposób następujący:

```
gs -sDEVICE=ljet2p -r300 -dNOPAUSE -sOutputFile=mój.hp mój.ps
```

dla drukarki HP LaserJet IIp. A dla drukarki igłowej Epson:

```
gs -sDEVICE=epson -r360x180 -dNOPAUSE -sOutputFile=mój.hp mój.ps
```

po czym drukowanie odbywa się z poziomu DOS `c:> copy/b mój.hp lpt1`

7.4 Plik /printcap

Daj polecenie `more /etc/printcap` i na ekranie pokaże się zawartość tego pliku, która może wyglądać np. tak:

```
# local dot printer Epson LQ850
dotb3|dotb3|localprinter:\
        :lp=/dev/ttyb:br#9600:\
        :ms=-parity,cs8,-cstopb,crtscts:\
        :sd=/var/spool/lpd:\
        :lf=/var/adm/lpd-errs:

##################################################################
# entry for a remote SPARCprinter NeWSprint printer
lp|SPARCprinter|SPARCprinter, a Sun SPARCprinter printer:\
lp=:rm=lksu:rp=SPARCprinter:mx#0:sd=/var/spool/SPARCprinter:\
lf=/var/spool/SPARCprinter/log:
```

Można z tego zbioru wyczytać, że lokalna drukarka jest drukarką igłową i wywołuje się przez nazwę *dotb3*. Inna drukarka SparcPrinter (laserowa, obsługiwana programem NeWSprint) jest wywoływana nazwą *SPARCprinter*.

Możesz mieć kłopoty z interpretacją zawartości tego zbioru, trzeba wtedy zapytać BWO (Bardzo Ważną Osobę).

Rozdział 8

Edytor tekstu vi

No cóż, nie da się tego uniknąć. Muszę choć kilka słów o edytorze vi (wi aj). Dlaczego tak wzdycham? Nie lubię tego edytora, zresztą nie ja jeden, ale do pewnych prac jest niezastąpiony. W końcu nie jest taki trudny do użycia, zwłaszcza jeśli ma się małą, podręczną "ściągę".

Jest to pełnoekranowy edytor możliwy do użycia na wszystkich typach terminali. Pełnoekranowy – oznacza to między innymi, że przywołany do edycji zbiór kopiowany jest do obszaru pamięci nazywanym buforem i wszelkie nasze działania wykonujemy na tej kopii. Dopiero zakończenie pracy spowoduje zamianę starej zawartości zbioru na nową. Stara zawartość wędruje do zbioru o nazwie zbliżonej do początkowej.

Pełnoekranowy – oznacza to także, dla niektórych, możliwość poruszania się kursorem po całym ekranie (w rzeczywistości można przewijać [scroll] ekrany co w konsekwencji daje możliwość poruszania się kursorem w obrębie całego edytowanego tekstu).

co złego ma vi?

Dlaczego więc vi budzi niechęć? Prawdopodobnie dlatego, że jego polecenia są jednoliterowe i nie bardzo się kojarzą ze skutkami. Trzeba zbyt dużo pamiętać o co trudno jeśli niezbyt często się używa.

Jak uruchomić edytor? Bardzo prosto:

```
% vi nazwa.zbioru
```

Jeśli zbiór już istnieje to zostanie skopiowany do bufora i udostępniony do edycji. Jeśli nie ma zbioru o podanej nazwie to zostanie otwarty nowy, pusty zbiór, gotowy do wypełnienia.

Wykonajmy ćwiczenie ze zbiorem tekstowym o nazwie `test.viaj`:

```
% vi test.viaj
```

Załóżmy, że taki zbiór już istnieje w aktualnym katalogu, program przejdzie więc do rzeczy, czyli otworzy plik do edycji. Na ekranie zniknie wszystko to co tam było (właściwie to zostanie przykryte nowym ekranem) a na ekranie pojawi się coś takiego:

```
Pierwsza linia tekstu,
druga linia,
trzecia,
itd.
~
~
~
~
"test.viaj" 4 lines 43 characters
```

Puste linie (edytor chce zająć cały ekran, a zbiór ma tylko cztery linie) są zaznaczone znakiem tyldy. Ostatnia linia na ekranie nie należy do zbioru, jest informacją o nim. Ta ostatnia linia za chwilę okaże się ważna.

8.1 Tryby pracy

Są dwa podstawowe tryby pracy edytora: tryb pisania tekstu i tryb wydawania (przyjmowania przez edtor) poleceń (te tryby są udręczeniem, program nie informuje, w sposób jawny, w jakim trybie w danej chwili się znajduje).

Zaraz po uruchomieniu edytora znajduje się on w trybie przyjmowania poleceń co oznacza, że przyciskanie klawiszy klawiatury nie powoduje "pisania" ale jest wydawaniem poleceń.

W szczególności, w tym trybie, możemy **poruszać kursorem po ekranie.** "Strzałki" na klawiaturze pozwalają przesuwać kursor w oczekiwanym kierunku (są także inne kombinacje klawiszy do przesuwania kursora ale o tych na końcu).

"wyjść z edytora"

Możemy też, będąc w trybie wydawania poleceń, zakończyć pracę z edytorem ("wyjść z edytora" jak to ładnie się mówi). Przyciśnięcie klawisza : (dwukropek) spowoduje przesunięcie (samoczynne oczywiście) kursora do ostatniej linii ekranu (zniknie to co tam było wcześniej czyli informacja o edytowanym zbiorze, pojawi się dwukropek) i system oczekuje na wpisanie komendy "po dwukropku" (z tego możesz się domyślić, że są co najmniej dwa typy poleceń: polecenia po dwukropku i jakieś inne). Jeśli teraz, po tym dwukropku w ostatniej linii ekranu uda Ci się postawić literę q (małe ku) i nacisnąć klawisz *Enter* to możesz oczekiwać, że program zamknie zbiór (pamiętaj, że nic takiego, poza otwarciem do edycji, z zbiorem nie wyczyniališmy) i na ekranie pojawi się znów to co było przed zabawą z `vi`.

Uff! tyle napisałem a nic z tego nie wynika (no może prawie nic). Myślę jednak, że interesujący jest także tryb pisania (dodawania, poprawiania, usuwania itp.). Wróćmy więc do chwili zaraz po uruchomieniu edytora.

Jak przejść do trybu pisania?

Przecież zaraz po otwarciu edytor znajduje się w trybie przyjmowania rozkazów. Ustaw kursor (już potrafisz, użyj "strzałek") w miejscu gdzie chcesz pisać i naciśnij a (klawisz a, może kojarzyć się z [append]). Nie zobaczysz tego a na ekranie, niestety, ale od tej chwili każde naciśnięcie klawisza będzie już

zostawiać ślad (nawet *Enter*, który daje przejście do nowej linii oraz dwukropek, który jest normalnym dwukropkiem a nie początkiem polecenia), jesteś w trybie pisania!

Jak wrócić do trybu wydawania poleceń?
Wystarczy przycisnąć klawisz *Escape* i już. Dla pewności można to zrobić więcej niż jeden raz. Od tej chwili, każde następne przyciśnięcie klawisza oznacza wydanie polecenia.

Jak już to wszystko napisałem to myślę, że może nawet polubię tego viaja. Chyba nie taki straszny, a użyteczny na pewno!

8.2 Podstawowe polecenia vi

Jeśli już umiesz przechodzić z jednego trybu pracy do drugiego i wypisywać polecenia w ostatniej linii, to do dalszej pracy wystarczy Ci ściągawka - spis poleceń.

Podstawowe polecenia vi

	Otwarcie zbioru do edycji
`vi` *nazwa.zbioru*	otwarcie lub utworzenie zbioru
`vi +18` *nazwa.zbioru*	otwarcie 18 linii zbioru
`vi -r` *nazwa.zbioru*	odtworzenie awaryjnie zamkniętego zbioru
`view` *nazwa.zbioru*	otwarcie bez możliwości zmian
	Ruchy kursorem
←↓→↑	jak widać
`h`	w lewo
`j`	do dołu
`k`	do góry
`l`	w prawo
`w`	w prawo o jedno słowo
`b`	w lewo o jedno słowo
`Return`	w dół jedną linię
`Spacja`	w prawo
`H`	na górę ekranu
`M`	na środek ekranu
`L`	na dół ekranu
	Wstawienie znaku lub linii
`a`	wstawienie znaku na prawo od kursora
`A`	wstawienie na końcu linii
`i`	wstawienie znaku na lewo od kursora
`I`	wstawienie znaku na początku linii
`o`	wstawienie nowej linii poniżej kursora
`O`	wstawienie nowej linii powyżej kursora

Podstawowe polecenia vi cd.	
	Zmiana tekstu
`cw`	zmiana słowa (lub jego części) na prawo od kursora
`c`	zmiana linii
`C`	zmiana części linii
`r`	zmiana znaku pod kursorem
`r` *Enter*	podział linii
`J`	połączenie linii z linią poniżej
`~`	zamiana duże–małe litery
`u`	cofnięcie ostatniego polecenia
`U`	cofnięcie wszystkich poleceń dla linii
`:u`	cofnięcie poprzedniego polecenia z ostatniej linii
	Kasowanie tekstu
`x`	usunięcie znaku pod kursorem
`dw`	usunięcie słowa
`dd`	usunięcie całej linii (tej z kursorem)
`D`	usunięcie części linii (na prawo od kursora)
`:5,10 d`	usunięcie linii 5 do 10
	Kopiowanie i przesuwanie tekstu
`yy`	kopia linii
`Y`	kopia linii
`P`	wstawienie skopiowanej linii poniżej kursora
`p`	wstawienie skopiowanej linii powyżej kursora
`:1,2 co 3`	skopiowanie linii 1 i 2 oraz wstawienie za linią 3
`:4,5 m 6`	przesunięcie linii 4 i 5 za linię 6
	Numerowanie linii
`:set nu`	włączenie wyświetlania numeru linii
`:set nonu`	wyłączenie wyświetlania numeru linii
	Odszukanie linii
`G`	leć do ostatniej linii w zbiorze
`21G`	leć do linii 21
	Odszukanie i zamiana tekstu
`/łańcuch`	znajdź ciąg znaków `łańcuch`
`?łańcuch`	szukaj w tył ciągu znaków `łańcuch`
`n`	znajdź następny raz
`:s/xyz/Ab/gc`	znajdź *xyz* i zamień na *Ab*
	Wstaw inny zbiór do tego zbioru
`r` *nazwa.zbioru*	wstaw (wczytaj) zbiór za kursorem
`:34 r` *nazwa.zbioru*	wstaw (wczytaj) zbiór za linią 34
	Zamknięcie edycji
`:w`	zapisz zmiany
`:w` *nazwa.zbioru*	zapisz bufor do nowego zbioru
`:wq`	zapisz zmiany i zakończ edycję
`ZZ`	zapisz zmiany i zakończ edycję
`:q!`	wyjdź z edytora, porzuć zmiany

Rozdział 9

Zbiory konfiguracyjne

Interakcyjne programy, które są często używane, takie jak interpreter poleceń (powłoka `csh`) [C shell], `mail`, edytor `vi`, mają własne zbiory konfiguracyjne ustalające ich parametry. Zbiór `.mailrc` dotyczący poczty został już omówiony w p. (5.6). Inne zbiory omówimy w tym rozdziale.

Wspólną cechą zbiorów kofiguracyjnych jest to, że są ukryte. Zwykłe polecenie `ls` listowania zawartości katalogu ich nie wykazuje. Dopiero polecenie `ls -a` wykaże[3.5] je. Ich nazwa zaczyna się kropką, stąd czasami są nazywane zbiorami "z kropką".

9.1 Plik `.cshrc`

Przy otwieraniu sesji roboczej i uruchamianiu powłoki, system w pierwszej kolejności sięga do zbioru konfiguracyjnego. W przypadku powłoki `csh` jest to zbiór `.cshrc`, w którym zawarte są ustawienia zmiennych systemowych.

Zbiór taki może wyglądać jak poniżej:

```
# @(#)Cshrc 1.4 90/10/04 SMI
######################################################################
#
#         .cshrc file
#
#         initial setup file for both interactive and noninteractive
#         C-Shells
#
######################################################################

#         set up search path

set lpath = ( )  # add directories for local commands                  1
if ( ${?mychoice} != 0 ) then                                          2
  if ( ${mychoice} == "openwin" ) then
    set lpath = ( /usr/openwin/bin/xview /usr/openwin/bin $lpath )
  endif
```

```
endif

set path = (. ~ /bin $lpath /usr/lang /usr/local /usr/bin/X11          3
/usr/X11/bin /usr/X11/lib /usr/X11/xmailtool /interviews/bin/SUN4
/usr/local/bin /usr/ucb /usr/bin /usr/etc /usr/local/lib/bin ) .

setenv MANPATH /usr/man:/usr/local/lib/man:/usr/X11/man                4
setenv FONTPATH /usr/lib/fonts                                         5
setenv TEXCONFIG /usr/local/lib/tex/ps                                 6
setenv XDVIFONTS /usr/local/lib/tex/fonts.lj/%d/%f.pk:.%f.pk           7
# setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:/opt/SUNWspro/lib
#       cd path

$set lcd = ( )  #  add parents of frequently used directories          8
$set cdpath = (.. ~ ~/bin ~/src $lcd)                                  9

#         set this for all shells

set noclobber                                                          10

#         aliases for all shells

alias cd            'cd \!*;echo $cwd'                                 11
alias cp            'cp -i'
alias mv            'mv -i'
alias del           'rm -i'
$alias rm            'rm -i'
alias pwd           'echo $cwd'
$umask 002

#      skip remaining setup if not an interactive shell

if ($?USER == 0 || $?prompt == 0) exit                                 12

#         settings  for interactive shells

set history=40                                                         13
set ignoreeof
$set notify
$set savehist=40
$set prompt="% "
$set prompt="'hostname'{'whoami'}\!: "
$set time=100

#         commands for interactive shells

$date                                                                  14
$pwd
```

```
#         other aliases
alias kto           'rusers -l'                                    15
#alias a            alias
alias h            'history \!* | head -39 | more'
#alias u            unalias

#alias list         cat
#alias lock          lockscreen
alias m            more
#alias mroe          more
#alias type         more

#alias .             'echo $cwd'
#alias ..            'set dot=$cwd;cd ..'
#alias ,             'cd $dot '

alias dir          ls -F
#alias pdw           'echo $cwd'
#alias la            'ls -a'
alias ll           ls -la
alias ls           ls -F

#alias pd           dirs
#alias po           popd
#alias pp           pushd

#alias +w            'chmod go+w'
#alias -w            'chmod go-w'
#alias x             'chmod +x'

#alias j             'jobs -l'

#alias bye           logout
#alias ciao          logout
#alias adios         logout

#alias psg           'ps -ax | grep \!* | grep -v grep'
#alias punt          kill

alias r            rlogin
#alias run           source

#alias lp1           'lpr -P1'
#alias lq1           'lpq -P1'
#alias lr1           'lprm -P1'
alias nptool /usr/newspr/bin/nptool
```

```
#alias sd          'screendump | rastrepl | lpr -v &'

#alias edit        textedit

#alias help        man
#alias key         'man -k'

#alias mkae        make
```

Poszczególne linie (numery z prawej strony) oznaczają:

Linia 1: Tworzy zmienną, która doda ścieżkę zawierającą nazwę katalogu z lokalnymi poleceniami (będzie dodana do zmiennej `path` z linii 3).

Linia 2: W przypadku korzystania z "okien" dodaje ścieżkę do binariów okien.

Linia 3: Ustala zmienną `path` ścieżkę dostępu do częściej używanych programów. Ta linia powinna być jedną linią. Dla potrzeb wydruku podzielono ją na trzy linie.

Linia 4: Zmienna systemowa z ścieżką dostępu do manuali systemowych.

Linia 5: Zmienna systemowa z ścieżką dostępu do krojów czcionek (w tym przypadku dla TEXa)

Linia 6 i 7: Zmienne systemowe właściwe dla TEXa

Linia 8: Tworzy zmienną systemową dla dodania ścieżki dostępu do katalogów, które są *rodzicami* często używanych katalogów. Dla przykładu, jeśli często zmieniamy katalog (`cd`) do `/usr/man/man1` to należy wpisać w nawiasy zmiennej `lcd` ścieżkę `/usr/man`. Ta ścieżka będzie dodana do zmiennej `cdpath` (patrz niżej).

Linia 9: Ustawia zmienną systemową `cdpath`. Daje to np. dla `..` tę możliwość, że znajdując się np. w katalogu `/usr/man/man1` przy przejściu do katalogu `/usr/man/cat1` dajemy polecenie `cd cat1` i to wystarcza. Należy zaznaczyć, że `cdpath` działa liniowo, tzn. jeśli zarówno w `~/bin` jak i `~/src` znajduje się katalog `local`, to polecenie `cd local` przeniesie nas do `~/bin/local`.

Linia 10: Zabezpieczenie przed omyłkowym skopiowaniem zbioru na już istniejący.

Linia 11: Kilka aliasów, czyli równoważników nazw. Zwróć uwagę, że nie wszystkie wpisane do zbioru są aktywne (linie zasłonięte znakiem `#`).

Linia 12: Test sprawdzający czy są ustawione zmienne o nazwie `USER` lub `prompt`. Jeśli żadna z nich nie jest włączona to powłoka nie jest interaktywna. Oszczędza czas przy używaniu zagnieżdżonych powłok.

Linia 13: Kilka opcji powłoki. Pierwsza zatrzyma w pamięci 40 ostatnich poleceń powłoki `C shell`, druga zabezpiecza przed mimowolnym zakończeniem sesji po *Ctrl-D*, ostatnia (`set time=100`) wyświetla statystykę poleceń zajmujących więcej niż 100 sek pracy CPU.

Linia 14: Wyświetla datę i czas (druga katalog roboczy) przy starcie powłoki `C shell`.

Linia 15: Blok aliasów, do wykorzystania.

9.2 Plik `.login`

```
# @(#)Login 1.11 90/10/04 SMI
#####################################################################
#
#         .login file
#
#         Read in after the .cshrc file when you log in.
#         Not read in for subsequent shells.  For setting up
#         terminal and global environment characteristics.
#
#####################################################################


#       terminal characteristics for remote terminals:

#       Leave lines for all but your remote terminal commented
#       out (or add a new line if your terminal does not appear).

if ($TERM != "sun") then                                              1
#eval 'tset -sQ -m dialup:?925 -m switch:?925 -m dumb:?925 $TERM'
#eval 'tset -sQ -m dialup:?h19 -m switch:?h19 -m dumb:?h19 $TERM'
#eval 'tset -sQ -m dialup:?mac -m switch:?mac -m dumb:?mac $TERM'
#eval 'tset -sQ -m dialup:?vt100 -m switch:?vt100 -m dumb:?vt100 $TERM'
endif

#         general terminal characteristics

#stty -crterase                                                       2
stty -tabs                                                           3
#stty crt                                                            4
stty erase '^h'                                                      5
#stty werase '^?                                                     6
stty kill '^['                                                       7
#stty new                                                            8
```

```
#         environment variables

#setenv EXINIT 'set sh=/bin/csh sw=4 ai report=2'                    9
#setenv MORE '-c'                                                   10
#setenv PRINTER lw                                                  11
setenv MANPATH /usr/lang/man:/usr/openwin/man:/usr/man              12

#         commands to perform at login

#w         # see who is logged in

#
#If possible, start the windows system. Give user a chance to bail out
#
if ( 'tty' != "/dev/console" || $TERM != "sun" ) then
  exit # leave user at regular C shell prompt
endif

echo ""
#click -n # click -n turns off key click

echo ""
switch( $mychoice )
case sunview:
unset mychoice
echo -n "starting SunView (Control-C to interrupt)"
sleep 5
# default sunview background looks best with pastels
sunview
clear # get rid of annoying cursor rectangle
logout # logout after leaving windows system
breaksw
#
case openwin:
unset mychoice
echo -n "starting OpenWindows (Control-C to interrupt)"
sleep 5
# setenv OPENWINHOME ??? TBD NYD XXX
/usr/openwin/bin/openwin
clear_colormap # get rid of annoying colourmap bug
clear # get rid of annoying cursor rectangle
        rm -rf /tmp/* /tmp/.*
logout # logout after leaving windows system
breaksw
```

```
#
default:
echo unknown \"mychoice\" \"$mychoice\" in .login
unset mychoice
exit # leave user at regular csh prompt
endsw
```

Niektóre linie (numery z prawej strony) wymagają dodatkowego objaśnienia:

Linia 1: Wykonuje polecenia między linią 1 a `endif` dla ustawienia odpowiedniego typu terminala w przypadku otwierania sesji "na terminal" np. przez modem lub `telnet` z PC. Należy wybrać odpowiadający nam typ terminala i odsłonić linię (przez skasowanie #). Można też dopisać charakterystykę innego terminala.

Linia 2: Przywiązuje funkcję kasowania znaków linii poleceń do klawisza *backspace* w ten sposób, że kasowane znaki pozostają widoczne, jednak nie są przekazywane do interpretera poleceń po naciśnięciu *Enter*.

Linia 3: Rozwija tabulacje do spacji przy wyświetlaniu na ekranie.

Linia 4: Ustawia standardową charakterystykę CRT.

Linia 5: Przywiązuje funkcję kasowania znaków linii poleceń do klawisza *backspace*.

Linia 6: Przypisuje funkcję kasowania słowa do klawisza *Del*

Linia 7: Przypisuje funkcję kasowania linii do klawisza *Esc*. Nie używaj tej opcji jeśli używasz także edytora vi.

Linia 8: Dla użycia nowej wersji terminala.

Linia 9: Ustawia parametry startowe edytora vi (jeśli nie ma zbioru konfiguracyjnego `~/.exrc`).

Linia 10: Zezwala poleceniu `more` przykryć ekran (zamiast przewijania).

Linia 11: Ustawia nazwę domyślnej drukarki[7.1] (w tym przypadku *lw*).

Linia 12: Ustala ścieżkę dostępu do manuali systemowych.

9.3 Plik .logout

Zwykle jest pusty, a szkoda bo wiele spraw powinniśmy załatwiać przed zamknięciem sesji roboczej. Poniżej propozycja małego skryptu (trzy linijki), likwidującego zbędne pliki w katalogu /tmp. Wiele takich przejściowych plików odkłada się bez wiedzy użytkownika. Po pewnym czasie dochodzi do "zapchania" się dysku. Operator systemu ma tutaj utrudnione zadanie, bo właścicielami zbiorów (naprawdę zbytecznych) są użytkownicy. Lepiej więc kasować je samemu. Aby nie obciążać pamięci można te trzy linijki wpisać do .logout.

```
unalias rm
cd /tmp
rm -rf *.*
```

9.4 Plik .profile

Czytany jako pierwszy w przypadku powłoki otwarcia sh (tej z $).

```
#
# @(#)local.profile 1.3 92/11/06 SMI
#
stty istrip
PATH=/usr/bin:/usr/ucb:/etc:.
export PATH
#
# If possible, start the windows system
#
if [ 'tty' = "/dev/console" -a "$TERM" = "sun" ] ; then
 if [ ${OPENWINHOME:-""} = "" ] ; then
    OPENWINHOME=/usr/openwin
    export OPENWINHOME
 fi

  echo ""
  echo "Starting OpenWindows in 5 seconds (type Ctrl-C to interrupt)"
  sleep 5
  echo ""
   $OPENWINHOME/bin/openwin
   clear      # get rid of annoying cursor rectangle
   exit       # logout after leaving windows system
fi
```

Spis treści