

# **Podstawy informatyki**

Elektrotechnika I rok

**Podstawy systemu operacyjnego UNIX**  
Instrukcja do ćwiczenia

## 1. SYSTEM UNIX – WSTĘP.

Początki systemu UNIX sięgają końca lat sześćdziesiątych i prac prowadzonych w Bell Laboratories przez Kena Thompsona i Dennisa Ritchiego. Na początku lat siedemdziesiątych została opublikowana specyfikacja systemu. Przez następne lata system UNIX był i jest nadal rozwijany i udoskonalany. Rozpowszechnianie kodu źródłowego systemu spowodowało popularyzację rozwiązań związanych z UNIX-em i doprowadziło do powstania całej rodziny systemów pochodnych oraz systemów-klonów, kompatybilnych, tworzonych od podstaw. Obecnie nazwa Unix jest nazwą zastrzeżoną dla organizacji Open Group. W powszechnym użyciu stanowi ona synonim systemów wywodzących się ze wspomnianego pierwotnego pnia oraz wszelkich systemów-klonów. Standaryzacja całej tej grupy systemów operacyjnych ujęta jest w dwie grupy norm – POSIX i Single Unix Specification, co zapewnia znaczną ich kompatybilność. W ramach systemów wywodzących się z pierwotnego opracowania Bell Labs wyróżnia się dwie pochodne grupy systemów unixowych: System V (dawne Bell Labs) oraz BSD (oparta na opracowaniach Uniwersytetu w Berkeley).

Najpopularniejsze obecnie wersje systemów klasy Unix to:

BSD UNIX (Berkeley Software Distribution),  
 System V (wersja UNIX z AT&T),  
 SCO UNIX (Santa Cruz Operation),  
 Solaris i SunOS (firmy Sun),  
 HP-UX (Hewlett-Packard),  
 Ultrix (DEC)  
 Linux (opracowany przez Linusa Torvaldsa),

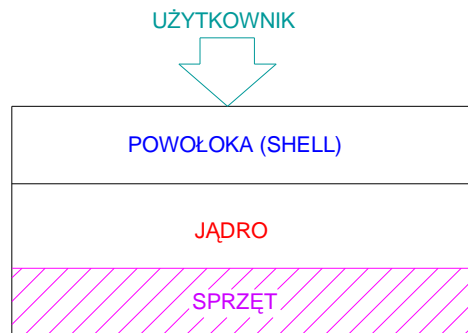
### 1.1 Charakterystyczne cechy systemu UNIX:

- Wielodostępność - równoczesna praca wielu użytkowników, przy zapewnieniu ochrony dostępu do komputera i prywatnych danych poszczególnych użytkowników,
- Wielozadaniowość - równocześnie może pracować wiele procesów (programów),
- Wielowątkowość,
- Hierarchiczny system plików.
- Zawiera wbudowane mechanizmy pracy sieciowej oraz rozproszonego systemu plików,
- Łatwo przenaszalny ze względu na dostępność kodów źródłowych systemu w języku C,
- Interaktywność – praca z systemem przy pomocy terminala,
- Wywłaszczanie.

Wywłaszczanie oznacza iż system operacyjny może przerwać wykonywanie procesu (wątku) (odebrać dostęp do procesora), przydzielając procesor innemu procesowi (wątkowi) na wykonywanie swoich zadań. Zatem system decyduje, jak długo dana aplikacja korzysta z procesora, i po upływie tego czasu następuje przekazanie kolejnej aplikacji procesorowi. (poprzednia zostaje wywłaszczona), lub jeśli dostępu do procesora zażąda proces o wyższym priorytecie. Dzięki temu jeśli pojedynczy program ulegnie zawieszeniu nie spowoduje zawieszenia całego systemu.

## 1.2 Struktura systemu UNIX.

System Unix ma budowę warstwową, którą przedstawia schematycznie rysunek 1.



Rys. 1 – Uproszczony schemat budowy systemu UNIX.

Pracę całego systemu koordynuje **jądro** (ang. kernel) – komunikuje się ono ze sprzętem pośrednicząc pomiędzy nim a procesami, zarządza przydziałem pamięci i czasu procesora i systemem plików. Żaden program nie ma bezpośredniego dostępu do urządzeń wej/wyj. Dostęp do jądra jest możliwy tylko przez wywołania funkcji systemowych (jest to charakterystyczna cecha systemu UNIX).

Między jądrem a użytkownikiem znajduje się **powłoka** (ang. shell) będąca interpretatorem poleceń użytkownika. Powłoka wywołuje także odpowiednie programy i zwraca ich wyniki wyprowadzane przez standardowe wyjście.

W systemach unixowych dostępnych jest kilka powłok, z których najbardziej popularne to:  
 Bourne Shell (sh),  
 C-Shell (csh)  
 Korn shell (ksh).

## 1.3 System plików

Każdy zbiór danych w systemach unixowych jest plikiem. Występują tu następujące rodzaje plików

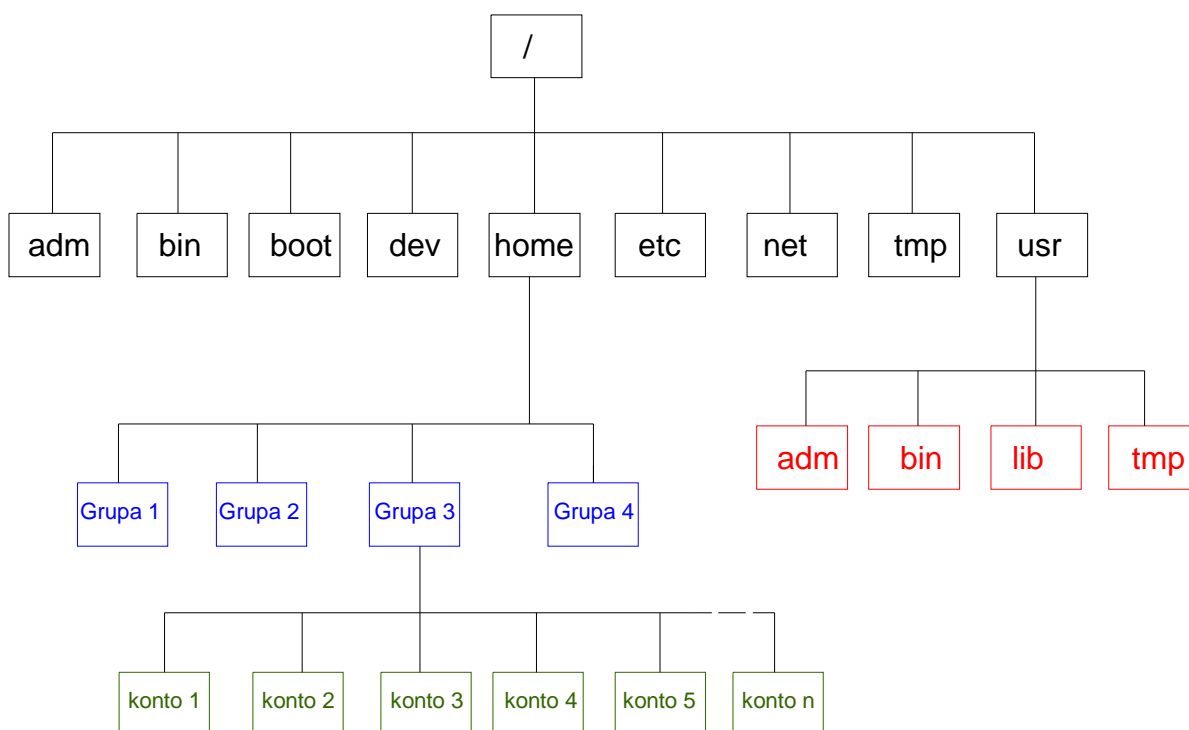
- Pliki normalne (ang. regular file) – zawierające różne dane,
- Katalogi (ang. directories) – zawierają nazwy innych plików oraz informacje o ich miejscu na dysku, prawach dostępu, itp.
- Pliki specjalne – reprezentują urządzenia znakowe komputera (ang. character special file) i urządzenia blokowe (ang. block special file). Dzięki temu mechanizmowi są widzialne w systemie plików jako zbiory.
- Kolejki FIFO (ang. fifo queue), pliki będące kanałami wejścia wyjścia,
- Skróty symboliczne (ang. symbolic link) - pliki zawierające nazwę ścieżki do innego pliku lub katalogu,
- Potoki – jednokierunkowe ograniczone bufory łączące standardowe wyjście jednego procesu ze standardowym wejściem innego.

Celem uporządkowania danych przechowywanych w systemie konieczne jest narzucenie struktury organizacyjnej informacjom zapisanych na fizycznych nośnikach (dyskach, taśmach, pamięciach reprogramowanych, itp.), które mogą być rozrzucone w różnych lokalizacjach. Aby ułatwić

odnalezienie poszukiwanych plików na podstawie nazwy, tworzone są katalogi (*ang. directories*), informujące o lokalizacji pojedynczych plików. Katalog plików zawiera nazwy plików na podstawie których sterownik określa ich fizyczną lokalizację na konkretnych nośnikach (dyskach).

Katalogi mogą zawierać również następne katalogi (nazywane podkatalogami). Tworzy to hierarchiczny, drzewiasty system plików charakterystyczny dla systemów unixowych. Pierwotny katalog w systemie plików nazywa się katalogiem głównym (*ang. root directory – katalog korzeń*). Zawiera on wszystkie pozostałe pliki w systemie. Katalog główny oznaczany jest znakiem ukośnika "/" i jego nazwy nie można zmienić

Drzewiasta struktura katalogów w systemie UNIX powoduje iż korzystanie z danego zbioru nie wymaga znajomości jego fizycznej lokalizacji, lecz położenia „logicznego” w strukturze. Typowa, uproszczona struktura systemu UNIX została przedstawiona na rysunku 2.



Rys. 2 - Typowa struktura katalogów w systemie UNIX.

- / – root, katalog główny,
- bin – katalog zawierający zbiory wykonywalne systemu (polecenia),
- boot – zbiór zawierający narzędzia do wytwarzania obrazu systemu,
- dev – pliki specjalne dla urządzeń (*ang. devices*) fizycznych (dyski, konsole, terminale, napędy CD-ROM, porty szeregowy, i inne),
- lib – biblioteki wywołań funkcji systemowych i bibliotek standardowych,
- etc – katalogi zawierające zbiory inicjalizacyjne i konfiguracyjne systemu,
- usr – katalog zawierający programy użytkowe,
- home – katalog zawierający konta użytkowników,
- tmp – katalog plików tymczasowych,

Każdy plik ma swoją nazwę, która może się składać z dowolnej kombinacji cyfr, liter i innych znaków alfanumerycznych, z wyjątkiem znaków mających znaczenie specjalne, jak:

& > < ! \* ? ] [ spacja tabulator @ # \$ ^ ( ) ' " ` | / \ ; ,

System rozróżnia duże i małe litery, na co należy zwracać szczególną uwagę, zwłaszcza przy przenoszeniu plików z systemu Windows.

Dostęp do pliku określa ścieżka dostępu czyli lista katalogów oddzielonych znakami '/' przez które należy przejść celem dotarcia do określonego pliku. Rozróżnia się ścieżki względne i bezwzględne.

Nazwa ścieżki bezwzględnej zaczyna się od katalogu głównego (znakiem „/”). Dzięki temu identyfikacja danego pliku jest jednoznaczna i niezależna od tego, w którym katalogu w danej chwili użytkownik się znajduje. Ścieżka względna identyfikuje dany zbiór z punktu widzenia katalogu bieżącego (w którym aktualnie użytkownik pracuje). Przykładem najkrótszej możliwej ścieżki względnej jest podanie samej nazwy pliku lub katalogu. W takim przypadku jest to ścieżka z katalogu bieżącego do podanego pliku lub katalogu, który się w danym katalogu bieżącym znajduje.

## 2. PRACA W SYSTEMIE UNIX

Każda osoba pracująca w systemie UNIX ma swoje konto, czyli przydzieloną część obszaru w systemie plików. Autoryzacja dostępu do konta jest możliwa za pomocą hasła. Każdy użytkownik należy do określonej grupy. Systemem zarządza administrator systemu, do którego uprawnień i obowiązków należy m.in. zarządzanie pracą systemu, przydział kont, identyfikatorów i haseł dla użytkowników i ich grupowanie.

Zakładając konto w systemie administrator określa m.in.:

- nazwę konta (login użytkownika)
- identyfikator (uid - user identification number),
- hasło,
- grupę (lub grupy) do jakich należy,
- katalog użytkownika (domowy),
- wielkość dostępnej pamięci dyskowej (quota).

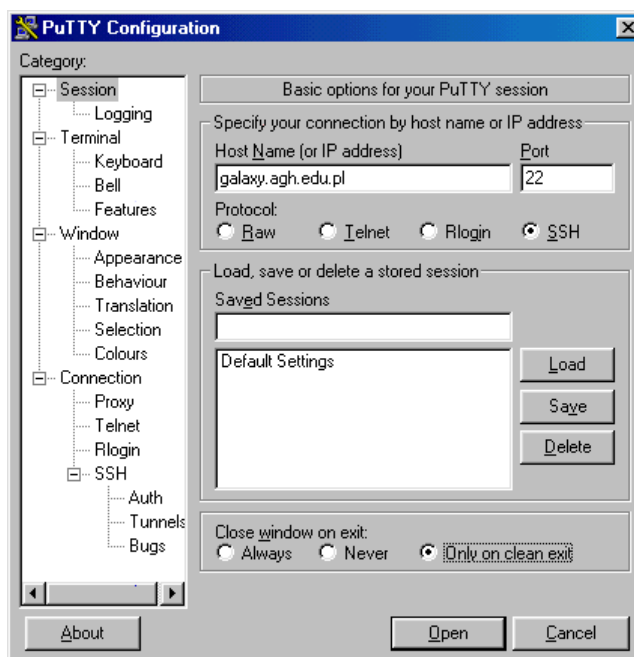
Katalog użytkownika (katalog domowy - ang. "home directory") jest to katalog zakładany podczas tworzenia konta każdemu użytkownikowi. Katalog ten należy tylko do tego danego użytkownika, mogącego tam umieszczać nowe pliki i tworzyć nowe katalogi. Pozostali użytkownicy (z wyjątkiem administratora) nie mają domyślnie pełnego dostępu do tego katalogu..

### 2.1 Rozpoczęcie pracy z systemem

Użytkownicy rozpoczynający pracę w systemie rozpoznawani są za pomocą nazwy (login) i hasła (password).

Najprostszym trybem pracy z systemem jest praca terminalowa, możliwa z każdego komputera funkcjonującego w sieci komputerowej. Korzystać można przykładowo z protokołu **Telnet** powszechnie używanego do obsługi odległych terminali. Istotną wadą takiego połączenia jest to, iż jest ono nieszyfrowane. Z tego względu często korzysta się z protokołów rodziny **SSH**, które wyparły inne bezpieczne protokoły (jak np: Rlogin, czy RSH)

Jednym z najpopularniejszych programów umożliwiających realizację połączeń szyfrowanych z komputerów pracujących w systemie Windows jest PuTTY. Obsługuje on protokoły SSH w pełni zastępując program telnet. Posiada duże możliwości konfiguracyjne oraz jest bardzo prosty w konfiguracji i użytkowaniu. Nie wymaga żadnej instalacji. Po ściągnięciu programu putty.exe i umieszczeniu we właściwym katalogu, można go uruchomić. Pojawi się wówczas okno konfiguracyjne "PuTTY Configuration", pokazane na następnej stronie.



Po lewej stronie okna widoczne są dostępne zakładki:

- Session,
  - o Logging,
- Terminal,
  - o Keyboard,
  - o Bell,
  - o Features,
- Window,
  - o Apperance,
  - o Behaviour,
  - o Translation,
  - o Selection,
  - o Colours,
- Connection,
  - o Telnet,
  - o Rlogin,
- SSH,
  - o Auth,
  - o Tunnels,
  - o Bugs.

Do nawiązania połączenia z komputerem należy wpisać nazwę komputera z którym chcemy się połączyć (np. galaxy.agh.edu.pl). Po nawiązaniu połączenia z komputerem należy zarejestrować się w systemie, co sygnalizowane jest z reguły przez system komunikatem **login:** należy podać nazwę użytkownika, zatwierdzić [Enter] i następnie system poprosi o hasło zgłaszając: **password:** . Podczas wpisywania hasła komputer nie wyświetla znaków, które się wpisuje. Należy pamiętać o tym że system Unix rozróżnia duże i małe litery. W przypadku błędnego podania identyfikatora lub hasła system wypisuje komunikat:

```
Login incorrect ' login:
```

Po poprawnym podaniu konta i hasła system zgłasza się do pracy znakiem zachęty, co oznacza że użytkownik jest zalogowany w systemie. Wszelkie czynności od chwili rozpoczęcia pracy z systemem aż do jej zakończenia nazywa się sesją.

## 2.2 Zakończenie sesji

W celu zakończenia sesji należy podać polecenie logout lub nacisnąć klawisze Ctrl + D. Następuje wtedy zakończenie połączenia z serwerem.

## 2.3 Wydawanie komend w systemie UNIX

Komunikacja użytkownika z systemem UNIX następuje poprzez wydawanie komend interpretowanych oraz realizowanych przez powłokę (shell). Składnia polecenia w shellu jest następująca:

**nazwa\_polecenia opcje argumenty**

*nazwa\_polecenia* - komenda dostępna w systemie lub własny program użytkownika (skrypt).

*opcje* - jedna lub kilka opcji dostępnych z daną komendą. Opcje modyfikują i precyzują wykonanie danego polecenia. Najczęściej podaje się je jako pojedyncze litery po znaku „-”.

*argumenty* - obiekty na których wykonywane są polecenia, np. pliki, urządzenia zewnętrzne.

## 3. PODSTAWOWE KOMENDY SYSTEMU UNIX.

### 3.1 Komendy organizacyjne

**passwd** - umożliwia zmianę hasła wpisywanego przez użytkownika przy logowaniu się do systemu. Na żądanie systemu należy podać kolejno stare hasło i dwukrotnie nowe.

**quota -v** - wyświetla wykorzystanie przydzielonego limitu dyskowego w obrębie konta.

**clear** - czyści ekran.

**logout, exit, CTRL + D** - zakończenie pracy

**date** - aktualna data i godzina,

**cal** – kalendarz,

**bc** - kalkulator

przykładowo:

```
bc <Enter> 23+71 <Enter>
```

**wyjście z systemu: Ctrl + d lub quit.**

**factor** – dokonuje podziału liczby na czynniki pierwsze i wyświetla je.

**env** – wyświetlenie informacji o środowisku (ang. environment), w którym pracuje użytkownik.

**tty** – podaje bieżący kod terminala,

**w, who** - wyświetla listę użytkowników aktualnie pracujących w systemie

**who am i** - wyświetla identyfikator użytkownika wpisującego tę komendę

**uname** – polecenie to powoduje wyświetlenie nazwy systemu operacyjnego. W połączeniu z opcją -a otrzymuje się kilka dodatkowych informacji na temat danego systemu operacyjnego, nazwy komputera.

**man** – polecenie wyświetla tekst pomocy dla danego polecenia. Składnia polecenia man jest następująca:

`man nazwa_polecenia`, gdzie *nazwa\_polecenia* jest nazwą polecenia systemowego.

Wyjście z opisu dokonuje się najczęściej komendą **q** (quit).

### 3.2 Znaki ogólnego zastosowania:

**&** - przenoszenie procesu w tło tj. uruchamianie dodatkowego okna obsługującego proces

**>, >>** - przekierowanie wyniku procesu,

**<** - pobranie danych do rozkazu,

**|** - rozdzielenie kilku poleceń zapisanych w jednym wierszu w tzw. potok.

### 3.3 Operacje na plikach i katalogach.

Po rozpoczęciu sesji użytkownik znajduje się w swoim katalogu domowym, który jest mu przypisany w momencie zakładania konta.

**ls** – polecenie (skrót od 'list') pozwala wylistować zawartość katalogu. Składnia polecenia jest następująca:

`ls [opcje][nazwa_katalogu / zbioru(ów)]`

Jeżeli nie podana jest nazwa katalogu lub zbioru, to wylistowany zostanie katalog bieżący.

Najczęściej stosowane *opcje* to:

-a wyświetlane są wszystkie zbiory (również te, których nazwa zaczyna się od znaku kropki).

-D wyświetlane są tylko katalogi

-l wyświetlanie informacji o poszczególnych zbiorach w danym katalogu w długim formacie (ang. long) – nazwy, atrybuty zbiorów, właściciel i grupa, czas modyfikacji, itp.

Poszczególne opcje mogą być łączone ze sobą, np.

`ls -al | more` - wyświetla listę plików, dzieląc je na ekrany

**pwd** – służy do sprawdzenia w jakim katalogu użytkownik znajduje się w danym momencie (jaki katalog jest katalogiem bieżącym).

**cd** – zmiana katalogu (ang. change directory) – umożliwia poruszanie się po katalogach. Składnia polecenia jest następująca:

`cd [opcje]`

`cd /` - przejście do katalogu podstawowego,

`cd ..` - przejście o jeden katalog w górę (w hierarchii drzewa katalogów),

`cd ../..` - przejście o dwa katalogi w górę,

`cd nazwakatalogu` - przejście do katalogu o wybranej nazwie

`cd` - przejście do katalogu domowego z dowolnego miejsca.

Przykłady:

`cd mucha <Enter>` - przejście do katalogu mucha znajdującego się w katalogu bieżącym

`cd /mucha <Enter>` - przejście do katalogu mucha znajdującego się w katalogu głównym systemu plików  
`cd .. <Enter>` - przejście do katalogu o jeden poziom wyżej

**find** – wyszukiwanie pliku  
`find /wkatalogu -szukana_nazwa`

**cat** – wyświetla zawartość pliku o wyspecyfikowanej nazwie pliku  
`cat nazwa_pliku`

**more** – filtr umożliwiający wyświetlanie zawartości pliku tekstowego na ekranie terminala (najczęściej monitorze komputera z którego łączymy się z serwerem). Umożliwia wstrzymanie wyświetlania po każdym wypełnieniu ekranu.  
`more nazwa_pliku`

**less** – podobnie jak powyżej, umożliwia także przeglądanie wstecz.  
`less nazwa_pliku`

**wc** – zliczanie liczby linii, słów lub znaków w pliku.

Składnia: `wc [opcje][nazwa_pliku]`

opcje:

-c – zliczanie znaków,

-l – zliczanie linii,

-w – zliczanie słów.

**mkdir** – polecenie to (skrót od „make directory”) pozwala na utworzenie nowego katalogu.

Składnia polecenia jest następująca:

`mkdir nazwa_katalogu,`

gdzie: `nazwa_katalogu` określa nazwę katalogu, który ma być utworzony.

Przykłady:

`mkdir Trzmiel <Enter>` - tworzy katalog o nazwie Trzmiel w katalogu bieżącym

`mkdir ../pszczola` - tworzy katalog o nazwie pszczola w katalogu o jeden poziom wyżej w stosunku do bieżącego katalogu.

**rmdir** - polecenie `rmdir` pozwala na usunięcie katalogu. Katalog musi być pusty. Składnia polecenia jest następująca:

`rmdir nazwa_katalogu`

**rm** – polecenie `rm` (skrót od „remove”) pozwala na usunięcie zbioru(ów) lub drzewa katalogów. Składnia polecenia `rm` jest następująca:

`rm [opcje] nazwa_zbioru(ów)`

gdzie `nazwa_zbioru(ów)` określa zbiory lub katalogi które mają być usunięte.

Najczęściej używane opcje to:

- i system żąda potwierdzenia usunięcia każdego zbioru
- f system nie żąda potwierdzenia usunięcia zbiorów które mają zabroniony zapis
- v w czasie usuwania są listowane usuwane zbiory
- R usuwanie całego drzewa katalogów łącznie z katalogami
- d usuwanie całego drzewa katalogów bez usuwania samej struktury katalogów

Przykłady:

- ```
rm MojZbior <Enter>      - usuwa zbiór o nazwie MojZbior z bieżącego katalogu
rm -v katalog/* <Enter> - usuwa wszystkie zbiory w katalogu katalog i listuje usuwane
                        zbiory

rm -vR* <Enter>         - usuwa wszystkie zbiory i katalogi z katalogu bieżącego
                        i listuje usuwane zbiory
```

**mv** – polecenie mv (od ang. „move”) służy do przenoszenia lub zmiany nazwy zbiorów. Składnia polecenia mv jest następująca:

```
mv [opcje] zbiór(y)_źródłowy(e) zbiór_docelowy,
```

gdzie: *zbiór\_źródłowy* określa nazwę zbioru(ów), który ma być skopiowany do miejsca określonego przez nazwę *zbiór\_docelowego*.

Na przykład:

```
mv jeden.txt dwa.txt      - kopiuje zbiór jeden.txt na zbiór dwa.txt.
```

**cp** - kopiuj (ang. copy) - polecenie to służy do kopiowania zbioru lub drzewa katalogów. Składnia polecenia jest następująca:

```
cp[opcje] zbiór(y)_źródłowy(e) zbiór_docelowy,
```

gdzie: *zbiór\_docelowy* określa nazwę zbioru lub katalogu docelowego do którego ma być skopiowany zbiór źródłowy. Jeżeli ma być skopiowanych kilka zbiorów, to miejscem docelowym musi być katalog.

Najczęściej używane opcje to:

- v w trakcie kopiowania są listowane zbiory kopiowane.
- R kopiowanie drzewa katalogów. Parametr ten musi być użyty jeśli zbiorem źródłowym jest katalog.
- i podczas kopiowania system zażąda potwierdzenia, jeżeli nastąpi zapisanie kopiowanego zbioru na już istniejącym.
- c w czasie kopiowania tworzone jest automatycznie drzewo katalogów, takie jak w katalogu źródłowym. Używana jest przeważnie łącznie z opcją -R

Przykłady:

- ```
cp jeden.txt dwa.txt <Enter> -kopiuje zbiór jeden.txt na zbiór dwa.txt
cp -v oko* kopia/ <Enter> -kopiuje wszystkie zbiory których nazwy zaczynają się
ciągiem znaków 'oko' do katalogu kopia i listuje nazwy zbiorów kopiowanych.
```

`cp -vRc * ../backup/` <Enter> -kopiuje wszystkie zbiory i katalogi z katalogu bieżącego do katalogu backup znajdującego się o jeden poziom wyżej w stosunku do bieżącego i listuje nazwy kopiowanych zbiorów.

**ln** – tworzenie linku (dowiązania) do pliku. Nowy plik może znajdować się w zupełnie innym katalogu niż plik macierzysty. Takie połączenie nazw z plikiem nosi nazwę linku, co można rozpoznać przy wyświetleniu zawartości katalogu po oznaczeniu **l** (ang. link) znajdującym się na pierwszym miejscu opisu typu pliku.

Rozróżnia się linki twarde i linki symboliczne. Link twardy to nadanie innej nazwy istniejącemu plikowi (przez to link oraz oryginał stają się przez to nierozróżnialne), natomiast link symboliczny jest specjalnym typem pliku wskazującym na plik oryginalny poprzez jego nazwę (jest to wpis w katalogu wskazującym na nazwę oryginalnego pliku).

Składnia polecenia **ln**, w przypadku tworzenia linku twardego do pliku `plik_a` wygląda następująco:  
`ln plik_a plik_b.`

Dodanie opcji **-s** pozwala na utworzenie linku symbolicznego.

### 3.4 Ochrona i prawa dostępu do plików

Ze względu na to, że Unix jest systemem wielodostępnym, dostęp do każdego pliku w systemie jest chroniony poprzez system własności plików i praw dostępu. Każdy plik ma swojego właściciela i jest przypisany do grupy użytkowników. Prawa dostępu podzielone są na trzy grupy: dostęp dla właściciela, grupy użytkowników i pozostałych, a przydzielane prawa to:

- r* - (*read*) prawo czytania,
- w* - (*write*) prawo zapisu,
- x* - (*execute*) prawo wykonywania pliku.

W przypadku katalogu prawo zapisu oznacza, iż zawarte w nim pliki można usuwać i zmieniać.

Przykładowy układ praw dostępu do katalogu:

<b>d</b>		<b>rwX</b>		<b>r-X</b>		<b>--X</b>
<i>typ pliku ~ właściciel ~ grupa ~ wszyscy</i>						

**d** - oznacza że dany zbiór jest katalogiem (ang. directory)

Zatem w tym przykładzie: właściciel ma prawo do *odczytu, zapisu i wejścia do katalogu*, grupa i wszyscy mają prawo *odczytu i wejścia do katalogu*

Ustalenie praw dostępu do plików jest podstawową formą ochrony plików w systemie. Oprócz właściciela plikami może dysponować tzw. użytkownik uprzywilejowany root, którym z założenia jest administrator systemu.

Do zmiany uprawnień użytkowników w stosunku do pliku czy katalogu służy polecenie **chmod** (*change mode* - zmień tryb). Składnia polecenia jest następująca:

`chmod ugo nazwa_pliku`

gdzie: u określa właściciela (ang. user), g- grupę w ramach której pracuje dany użytkownik (ang. group), natomiast o –wszystkich innych (ang. other)

Prawa dostępu można podawać w dwojaki sposób:

- w postaci liczbowej z zastosowaniem zapisu ósemkowego, przy założeniu, że prawu odczytu r odpowiada wartość 4, prawu zapisu w - 2, a prawu wykonywania x – 1, brak danego prawa równoważny jest 0. W ramach danej grupy użytkowników wartości liczbowe sumują się.
- w postaci specyfikacji: u [+ | -] [rwx] , g [+ | -] [rwx] , o [+ | -] [rwx] (dodajemy (znak +), lub odbieramy (znak -) dane prawo).

Oznacza to iż komendy:

```
chmod u=rwx,g=rx,o=r plik
chmod 754 plik
```

ustawiają te same prawa dostępu, czyli właściciel pliku *plik* będzie miał wszystkie uprawnienia (rwx, czyli liczbowo  $4+2+1=7$ , stąd pierwsza liczba 7 dla użytkownika), członkowie grupy będą mogli *wykonać go lub odczytać* (r-x, czyli  $4+0+1=5$ ), a inni mogą *go tylko odczytać*.

### 3.5 Narzędzia do obróbki tekstu i inne programy użytkowe.

- **edytor vi.**

Edytor vi jest powszechnie dostępny we wszystkich odmianach systemu UNIX. Wykorzystywany jest najczęściej do pisania skryptów i programów. Aby utworzyć lub edytować plik należy użyć wywołania **vi** z nazwą pliku jako argumentem.

```
vi nazwa_pliku
```

podstawowe funkcje edytora:

i - przejście w tryb edytowania (pisanie tekstu; przejście do następnego wiersza [Enter];poprawki tylko w trybie rozkazów)

<Esc> - przejście do trybu rozkazów

x - kasowanie znaku znajdującego się przy kursorze,

dw - kasowanie wyrazy w którym znajduje się kursor,

dd - kasowanie wiersza w którym znajduje się kursor,

:w – zapis pliku na dysk,

:q! - wyjście bez zapisu,

:x - wyjście z zapisaniem pliku,

:q – wyjście z edytora, jeśli plik nie był modyfikowany.

- **kompilator języka C.**

Do uruchamiania kompilatora systemowego służy polecenie **cc** (ang. C compiler). Uruchamia ono kompilator systemowy (kompilator C, C++, asemblera, itd.) i ewentualnie linker. Składnia polecenia jest następująca:

```
cc [opcje] nazwa_zbioru
```

Gdzie *nazwa\_zbioru* jest nazwą zbioru w danym języku, który ma zostać poddany kompilacji.

Przykłady:

```
cc program.cpp – następuje kompilacja (bez konsolidacji) programu program.c
z zapisaniem kodu wynikowego w pliku a.out
```

```
cc -o wynik program.cpp – jak wyżej, ale z zapisem kodu wynikowego w pliku wynik.out
```

### 3.6 Procesy

Proces w uproszczeniu oznacza uruchomiony program. Podstawowym poleceniem dzięki któremu można uzyskać informacje o procesach jest **ps**.

**ps** – listuje procesy aktualnie uruchomione.

Bez dodatkowych opcji powoduje ono wyświetlenie listy procesów przypisanych do terminala z którego wywołane zostało przez użytkownika to polecenie. Warto stosować to polecenie przed zakończeniem pracy. Najważniejsze informacje pojawiające się po wywołaniu tego polecenia to numery PID oraz TTY.

Numer PID oznacza identyfikator procesu (ang. process identification number). Jest to unikalny numer przypisywany każdemu procesowi przez jądro w momencie powstania danego procesu.

Numer TTY (ang. teletype – ang. dalekopis) oznacza nazwę pliku oznaczającego terminal, przy którym pracuje dany użytkownik.

W każdym przypadku proces można usunąć poleceniem **kill**. Składnia polecenia jest następująca:

```
kill identyfikator_procesu
```

gdzie: identyfikator\_procesu oznacza wspomniany powyżej numer PID.

### 3.7 Strumienie i potoki.

Strumienie danych to operacje wejścia/ wyjścia umożliwiające wymianę danych. Każdy z procesów uruchamianych w systemach unixowych ma związane ze sobą trzy strumienie (widoczne oczywiście jako pliki):

- standardowe wejście (stdin), zwykle przyporządkowane do klawiatury. Plik ten jest oznaczany numerem 0.
- standardowe wyjście (stdout) , jest zwykle przyporządkowane do monitora. Do tego pliku procesy zapisują efekty swojego działania, i dlatego pojawiają się one na ekranie monitora. Jest oznaczany numerem 1.
- standardowe wyjście sygnalizacji błędów (stderr), podobnie jak *stdout* jest zwykle przyporządkowany do monitora, przez co komunikaty błędów widzimy więc na ekranie. Pliki stdout i stderr są zupełnie odrębnymi plikami. Plik stderr jest oznaczany numerem 2.

Przyporządkowanie każdego z tych trzech wymienionych plików można zmienić. Przekierowanie wyjścia jednego programu na wejście drugiego następuje poprzez tzw. **potoki** (ang. *pipeline*). Dokonuje się tego za pomocą użycia znaków przekierowania <, > oraz >>.

- **przekierowanie wyjścia.**

Przekierowanie wyjścia danego procesu można uzyskać po zatwierdzeniu komendy:

```
nazwa_polecenia > plik
```

Uruchomiony zostaje proces nazwa\_polecenia i wyniki swojego działania zapisze on nie na ekran monitora, ale do pliku **plik**. Ponowne użycie tego samego pliku jako wyjścia do którego zapisywane są wyniki, ze znakiem przekierowania „>” powoduje zastąpienie starej zawartości pliku nową. W niektórych przypadkach może to być niewskazane, zatem aby uniknąć ponownego zapisywania, należy zamiast znaku „>” zastosować „>>” co spowoduje dopisanie nowej zawartości (pochodzącej z procesu) do już istniejącej.

Przykład:

```
ls -a >> katalogi.txt
```

spowoduje zapisanie nazw wszystkich katalogów znajdujących się w katalogu bieżącym do pliku katalogi.txt.

- **przekierowanie wejścia.**

Przyporządkowanie wejścia dokonywane jest z zastosowaniem znaku „<”. Załóżmy że chcemy przekierować zawartość pliku katalogi.txt

```
cat < katalogi.txt
```

program uruchomiony polecenie cat przyjmie za dane wejściowe zawartość pliku katalogi.txt i wyświetli jego zawartość na ekranie.

Można też połączyć oba typy przyporządkowań, rozdzielając je średnikiem w linii komend shella, na przykład:

```
ls -a > katalogi.txt
```

```
wc -l < katalogi.txt
```

wynik wyświetlenia wszystkich katalogów zostanie zapisany w pliku katalogi.txt a następnie ilość linii w tym pliku zostanie zliczona za pomocą wywołania polecenia wc.

- **łączenie wejść i wyjść procesów**

Tworzenie potoków polega na kierowaniu wyników działania jednego programu (zamiast do wyjścia standardowego) na wejście standardowe kolejnego programu.

Przykładowo:

```
ls |sort
```

wyświetli nam posortowana listę plików w danym katalogu (bez podania nazwy – w katalogu bieżącym)

Łączenie to zostało już zastosowane w p. 3.3 przy omawianiu polecenia ls.

### 3.8 Interakcje pomiędzy użytkownikami

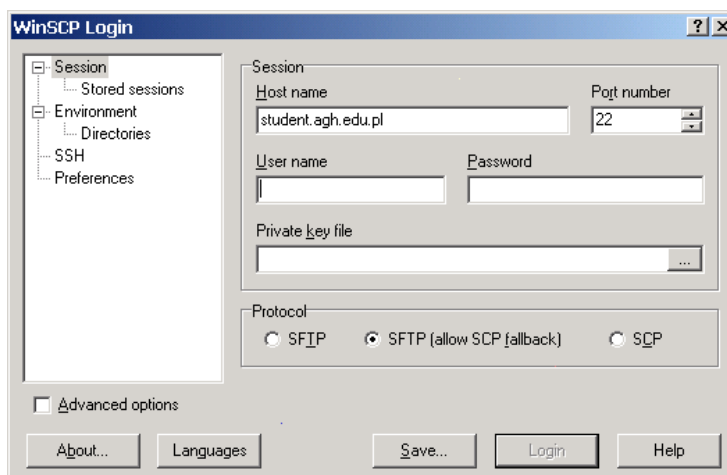
**finger** - pokazuje aktywne konta na wybranym serwerze,  
**talk** - interakcyjne porozumiewanie się.

### 3.9 Przenoszenie danych z komputera z Windows 95/98 na serwer UNIX – WinSCP.

Często w trybie pracy terminalowej jako terminal używany jest komputer z zainstalowanym systemem operacyjnym Windows. W przypadku potrzeby przenoszenia plików z danymi z tego komputera (z jego dysku lokalnego, ze stacji dyskiety) warto posłużyć się programami ułatwiającymi przeprowadzenie takiej operacji. Jednym z nich jest WinSCP. Jest to darmowy program, pełniący rolę klienta SCP (Secure CoPy) dla systemów Windows 95/98/2000/NT. Jest on oparty na protokole SSH. Głównym zadaniem programu WinSCP jest zapewnienie bezpiecznego kopiowania plików pomiędzy komputerami: lokalnym i zdalnym. Poza tym program realizuje szereg innych operacji na plikach (przenoszenie, zmiana nazwy, itp.).

Uruchomienie programu WinSCP najpierw wywołuje okienko zgłoszenia (WinSCPLogin – jak na rysunku poniżej), a następnie panel główny programu. Okienko Login składa się z czterech pól tekstowych, w które należy wpisać odpowiednio:

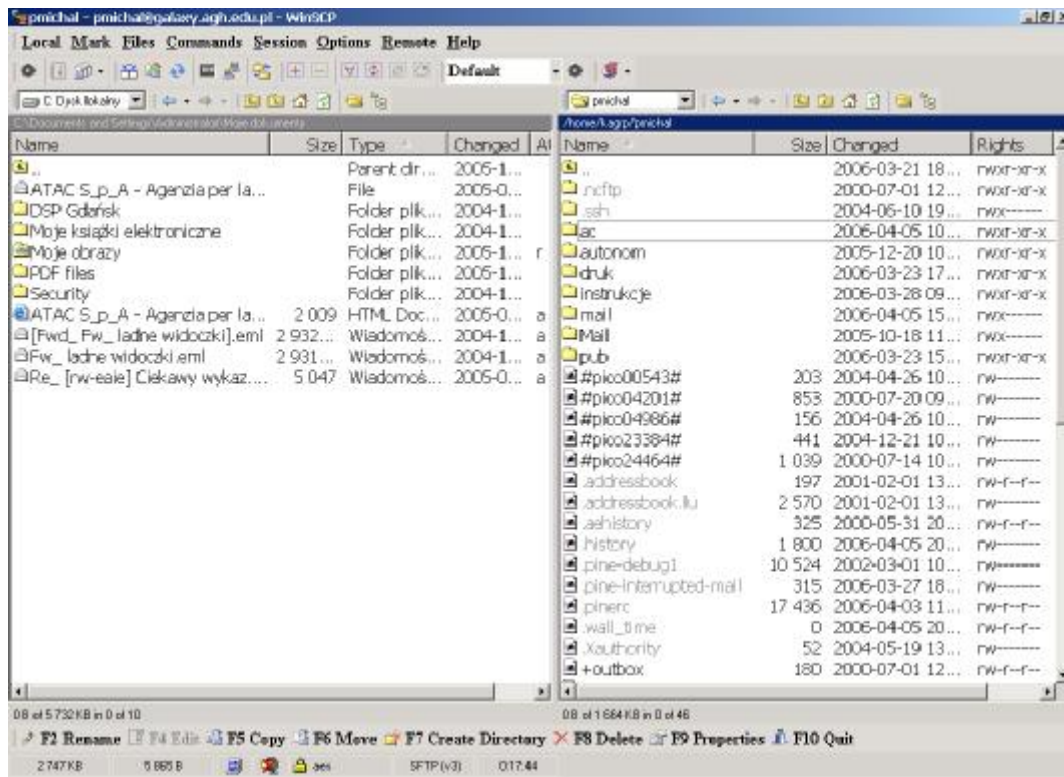
Enter host:	nazwę serwera, z którym chcemy się połączyć,
Enter username:	swój login na tym serwerze,
Enter password:	i odpowiadające mu hasło,
Enter port:	numer portu, dla SSH jest nim 22.



Po prawidłowym zalogowaniu użytkownika pojawia się okno programu WinSCP. Składa się ono z dwóch okien bazuje na układzie interfejsu znanego z programu Norton Commander. Katalogi i pliki komputera lokalnego są wyświetlane w lewym oknie, a komputera zdalnego w prawym. Sposób wyświetlania listy plików jest taki sam jak w Windows Explorer.

Program zawiera wiele funkcji, m.in. dotyczących obsługi plików i katalogów - podgląd, edycja, kombinacji klawiszy Alt - F1 (dla panelu lewego) lub Alt - F2 (dla panelu prawego). WinSCP może wykonywać wszystkie podstawowe operacje na plikach, np.:

- kopiowanie do/ze zdalnego komputera,
- usuwanie plików i katalogów,
- zmiana nazwy plików i katalogów,
- tworzenie nowych katalogów na lokalnym i zdalnym,
- zmiana uprawnień i przynależności do grup użytkowników dla plików i katalogów.



Wymienione wyżej działania można wykonywać w sposób znany z Norton Commandera. Są one realizowane wyłącznie w aktywnym oknie. Przełączanie między oknami następuje klawiszem **Tab**. Dostępne operacje posiadają własne skróty klawiszowe (np. zmiana nazwy F2, kopiowanie F5, usuwanie F8, itp. ). Zakończenie pracy w programie następuje za pomocą klawisza F10.

#### 4. Program ćwiczenia.

Program ćwiczenia obejmuje:

- zapoznanie się z poprawnym rozpoczynaniem i kończeniem pracy w systemie,
- podstawowymi komendami sieciowymi,
- zapoznanie się ze strukturą plików i katalogów w systemie, poruszaniem się po katalogach, podstawowymi operacjami związanymi z plikami i katalogami,
- mechanizmy ochrony plików – prawa dostępu,
- pojęcie strumienia i potoki wykorzystanie oraz podstawowe komendy z nimi związane,
- podstawowe aplikacje użytkowe w systemie,
- kopiowanie plików i katalogów z komputera lokalnego na konto użytkownika,

#### 5. Literatura.

- M. Kaniewski, K. Wieremiejczyk – Po prostu UNIX, Zakład Nauczania Informatyki MIKOM, Warszawa, 1992.
- Z. Królikowski – Użytkowanie systemu operacyjnego UNIX, Wydawnictwo Politechniki Poznańskiej, Poznań, 1992.
- W. Moczuard – W sieci - UNIX, sieci lokalne i rozległe, Wydawnictwo Fortis, Kraków 1993.
- M.J. Rochkind. – Programowanie w systemie UNIX dla zaawansowanych. WNT, Warszawa, 1993.
- P.P. Silvester. – System operacyjny UNIX. WNT, Warszawa, 1990.
- W.R. Stevens. – UNIX Programowanie usług sieciowych. Tom 1 API: gniazda i XTI. Tom 2 Komunikacja międzyprocesowa. WNT, Warszawa, 2000, 2001.