Podstawy pracy z systemem UNIX

Większość współczesnych komputerów (i podobnych urządzeń np. tablety czy telefony) wyposażonych jest w złożone oprogramowanie, które składa się na system operacyjny. Z całą pewnością korzystałeś z systemów operacyjnych firmy Microsoft – rodziny Windows. Mogłeś też zetknąć się z systemem Android (czyli odmianą Uniksa) od Google czy iOS od Apple. W większości przypadków system posiada tzw. interfejs graficzny czyli GUI (Graphical User Interface). Systemy te są zasadniczo podobne i np. uruchomienie przeglądarki internetowej czy przeglądanie dysku nie jest dla nikogo wyzwaniem.

Jednak nie każdy komputer posiada GUI, dotyczy to np. rozbudowanych komputerów wykorzystywanych w obliczeniach numerycznych (np. wyznaczanie właściwości aerodynamicznych samochodu z użyciem programu Fluent). W takim przypadku nie ma możliwości skorzystania z myszki i obejrzenia czegoś na ekranie, ponieważ komputer znajduje się w serwerowni, czasami w innym kraju. Aby korzystać z takiego zdalnego komputera musimy połączyć się z nim za pomocą specjalnego programu, który pozwoli nam na wydawanie mu polecenia w trybie tekstowym.

Na potrzeby tego laboratorium każdy otrzymał kartkę z loginem i hasłem. Są one ważne do końca semestru i można za ich pomocą zalogować się na nasz szkolny serwer `info3.meil.pw.edu.pl` również spoza kampusu.

Jeśli korzystasz z oprogramowania Windows, do połączenia najwygodniej wykorzystać darmowy program PuTTy. Po uruchomieniu należy podać dane:

- `Host Name`– nazwa hosta (`info3.meil.pw.edu.pl`),
- `Port`– Numer portu, z których chcemy się połączyć (22),
- `Connection Type` – typ połączenia (`SSH`).

Po kliknięciu `Open` pojawi się czarne okno z zapytaniem o login a następnie o hasło. Uwaga: znaki wpisywanego hasła nie są w żaden sposób zaznaczone (np. gwiazdkami) i jest ono niewidoczne. Po zalogowaniu się zobaczysz informacje o dacie, licencji, wersji systemu, itd. kończące się:

```
Last login: Thu Feb 21 06:23:38 2013 from xx.xx.xx.xx
stud01@eto:~$
```

W przypadku logowania się z systemu Linuks korzystamy z polecenia `ssh`:

```
ssh stud01@info3.meil.pw.edu.pl
```

Port 22 jest domyślnym portem używanym przez polecenie, nie trzeba go podawać explicite.

Zapis:

```
stud01@eto:~$
```

to tzw. znak zachęty i oznacza, że jako użytkownik `stud01` jesteśmy zalogowani na komputer `eto`. Między znakami `:` i `$` znajduje się katalog, w którym się aktualnie znajdujemy. W tym przypadku jesteśmy katalogu domowym. Znak `~` to skrót, którego rozwinięcie to `/home/students/stud01`.

Ćwiczenia

Pierwsze starcie

Wpisz w konsoli polecenie `date` i wciśnij enter. Komputer wyświetli aktualną (jego zdaniem) datę i godzinę. Poniżej ponownie wyświetli się linijka kończącą się na `$`, oznaczającą, że komputer czeka na nowe polecenia. Używając strzałek `góra` i `dół` możesz przeglądać historię poleceń. Kliknięcie klawiszy `Ctrl + R` uruchomi opcję wyszukiwania poleceń w historii. Klasyczny znak zachęty zostanie zastąpiony przez

```
(reverse-i-search)`':
```

Wyszukujemy przez wpisywanie kolejnych znaków z szukanego polecenia, a pod dwukropku system pokaże podpowiedź. Kolejne kliknięcie `Ctrl + R` pokaże następną sugestię. Innym ułatwieniem jest kończenie nazw. Jeśli wpiszesz `dat` i naciśniesz 2x klawisz `tab`, wyświetlona zostanie lista poleceń zaczynających się na `dat`. Jeśli jest tylko jedno takie polecenie, nazwa zostanie dokończona. Pamiętaj o tych trikach – znacznie ułatwiają pracę w trybie tekstowym.

Poruszanie się po katalogach

Pracując w trybie tekstowym, zawsze pracujemy w jakimś katalogu, tzw. katalogu bieżącym. Jeśli uruchomimy jakiś program, np. prosty program czytający dane z pliku z Informatyki I, będzie on odczytywał pliki znajdujące się w tym katalogu. Każdy program, którego będziesz używać, a który potrzebuje nazwy pliku lub katalogu (np. do kopiowania), może ją otrzymać w dwóch postaciach. Pierwsza to tzw. ścieżka bezwzględna, zaczynająca się od znaku / np:

```
/home/students/stud01
/usr/bin/bash
/etc
```

Sprawdź, w jakim katalogu się znajdujesz przez wpisanie polecenia `pwd`.

Aby zmienić katalog, wykorzystuje się polecenie `cd`, np.

```
stud01@eto:~$ cd /tmp
stud01@eto:/tmp$ pwd
/tmp
stud01@eto:~$
```

Teraz przejdź do katalogu `/home` i sprawdź czy się udało, z użyciem polecenia `pwd`. Aby powrócić do katalogu domowego wpisz

```
stud01@eto:~$ cd ~
```

Znak ~ zawsze oznacza katalog domowy użytkownika.

Dodatkowo, oprócz ścieżki bezwzględnej, można podać ścieżkę względną:

- `../` oznacza katalog nadrzędny,
- `/` oznacza katalog główny (początek każdej ścieżki bezwzględnej),
- `.` i `./` oznaczają katalog bieżący (ten zwracany przez polecenie `pwd`).

Poeksperymentuj teraz z poruszaniem się po katalogach. Jeśli wpisywanie ścieżek Cię znudzi wypróbuj program `mc`. Pozwala on m. in. na graficzne poruszanie się po drzewie katalogów. Z programu wychodzimy przez kliknięcie klawisza `F10` lub wpisanie `exit`.

Tworzenie i usuwanie katalogów

Do tworzenia katalogów służy polecenie `mkdir` np.

```
$ mkdir nazwa_katalogu
```

a do sprawdzenia zawartości aktualnego katalogu polecenie `ls`. Stwórz teraz katalogi A, B, C i D, każdy wewnątrz poprzedniego. Aby to zrobić będziesz musiał stworzyć katalog A, przejść do niego, następnie stworzyć B, itd. Do usuwania katalogów służy polecenie `rmdir`. Usuń teraz stworzone katalogi.

Za pomocą `rmdir` nie można usunąć katalogu posiadającego zawartość. W tym celu należy wykorzystać polecenie `rm -r`. Znaki `-r` po nazwie programu `rm` są argumentem programu i oznaczają, że katalog ma być usuwany rekurencyjne. Podobne argumenty posiada większość poleceń, np. `ls -l` pokazuje zawartość danego katalogu w postaci listy zawierającej różne informacje o pliku.

Podstawowe operacja na plikach i katalogach

Komenda `echo` wypisuje na ekran ciąg znaków, który podany jest jako jej argument. Można to wykorzystać do stworzenia pierwszego pliku (o znaczeniu symbolu `>>` będzie na kolejnych zajęciach)

```
$ echo pierwszy plik >> plik.txt
```

Aby wyświetlić zawartość pliku na ekranie używamy polecenia `cat`

```
$ cat plik.txt
```

Kopiowanie i przenoszenie

Do kopiowania służy komenda `cp CO GDZIE`. Stwórz teraz katalog i skopiuj do niego Twój plik. Powinno to wyglądać tak:

```
$ cp plik.txt katalog
```

Aby przenieść/zmienić nazwę pliku lub katalogu używamy polecenia `mv CO GDZIE`. Przejdź do nowego katalogu i zmień nazwę pliku. Następnie usuń plik poleceniem `rm`.

Nie zawsze trzeba podawać pełną nazwę pliku/katalogu, który chcemy wykorzystać jako argument programu. Kliknięcie klawisza `tab` dokończy wpisywaną nazwę. Jeśli podpowiedź nie jest jednoznaczna, po dwukrotnym naciśnięciu klawisza `tab` w konsoli zostaną wyświetlone nazwy wszystkich plików/katalogów zaczynające się od wpisanych znaków.

Pomoc

Znakomita większość komend trybu tekstowego posiada porządną dokumentację dostępną od ręki, np.:

```
$ man rm
$ rm --help
```

W przypadku komendy `man` dostajemy obszerniejszą dokumentację. Tekst przewija się za pomocą strzałek. Aby zakończyć przeglądanie należy wcisnąć `Q`. Sprawdź instrukcje dla poleceń `who`, `whoami`, `finger` i `date`. Sprawdź jak działają.

Program Tar

Program `tar` służy do pakowania i rozpakowywania drzewa katalogów i plików w jeden plik. Plik wynikowy niekoniecznie musi być mniejszy niż oryginalne pliki. Dopiero użycie kompresji zmniejszy rozmiar. Przygotuj najpierw kilka plików do spakowania:

```
$ mkdir a
$ cd a
$ mkdir b
$ echo asdasd >> ./b/c
$ cat ./b/c
asdasd
```

Teraz spakuj je a następnie obejrzyj zawartość archiwum za pomocą programu `mc`.

```
$ tar -cf test.tar b
$ ls
b  test.tar
$ mc
```

Sprawdź zawartość katalogu, który spakowałeś, następnie usuń go i rozpakuj archiwum.

```
$ ls
b  test.tar
$ rm -r b
$ ls
test.tar
$ tar -xf test.tar
$ ls
b  test.tar
```

Sprawdź poleceniem `ls -la` objętość utworzonego archiwum. Następnie spakuj te same pliki z dodatkową flagą `z`. Dodanie tego parametru uruchamia kompresję programem `gzip`. Zmień rozszerzenie archiwum na `tar.gz`. Sprawdź czy plik wynikowy jest mniejszy.

Proste skrypty

Najważniejszym aspektem pracy w trybie tekstowym jest możliwość tworzenia skryptów, czyli zapisanych w pliku kolejnych komend wykonywanych tak, jakbyśmy wpisywali je z klawiatury. Więcej o zaawansowanych skryptach dowiesz się na następnych laboratoriach, pierwszy napiszesz dzisiaj.

Prostymi i dość wygodnymi edytorami tekstu są `nano` i `vim`. Uruchom program `nano` komendą `nano skrypt.sh` i zapisz do niego pierwszy skrypt:

```
#!/bin/bash
echo 1
echo 2
```

Następnie zmień uprawnienia, aby pozwolić na uruchomienie naszego skryptu:

```
$ chmod u+x skrypt.sh
$ ./skrypt.sh
```

Polecenie `chmod` służy do zmiany uprawnień pliku. `u` oznacza użytkownika (czyli Ciebie), któremu chcemy nadać `+` prawo uruchamiania `x` skryptów/programów.

Zmienne

Bash obsługuje zmienne, tak jak język C. Nie występuje tu jednak typowanie. Aby stworzyć zmienną zawierającą tekst piszemy:

```
zmienna="Tekst"
```

Natomiast wynik działania jakiegoś programu możemy zapisać do zmiennej w następujący sposób:

```
zmienna=$( pwd )
```

Aby odczytać zmienną piszemy:

```
echo $zmienna
```

Drugi skrypt

Przygotuj strukturę katalogów:

- AA
  - BB
    - plik.txt
  - CC
    - DD
    - plik.txt

zawierając niezbędne komendy w skrypcie. Plik `plik.txt` ma zawierać datę zwracaną przez polecenie `date`.

Zmodyfikuj skrypt tak, aby nazwa każdego katalogu zaczynała się przedrostkiem przekazanym do skryptu przy jego uruchomieniu.

- przed_AA
  - przed_BB
    - plik.txt
  - przed_CC
    - przed_DD
    - plik.txt

Abu uruchomić skrypt wraz z dwoma argumentami `arg1` i `arg2` należy wpisać

```
./skrypt.sh arg1 arg2
```

Do argumentów `arg1` i `arg2` mamy dostęp z wewnątrz skryptu dzięki tzw. zmiennym programowym. Jest ich dziewięć `$1` – `$9`, w tym przypadku będą to zmienne `$1` i `$2`.

Pętle

Przygotuj skrypt zawierający:

```
#!/bin/bash

for i in *.txt
do
  cp $i $1_$i
done
```

i uruchom go w katalogu zawierającym pliki z rozszerzeniem `.txt`. W jaki sposób działa? Pamiętaj o argumencie skryptu!

Napisz skrypt, który tworzy katalog o nazwie podanej jako pierwszy argument skryptu `$1` i kopiuje do niego wszystkie pliki `.txt` dodając do ich nazwy przedrostek podany jako drugi argument `$2`. Co się stanie jeśli nie podasz argumentów do skryptu?

GUI

Jeśli korzystasz z systemu Linux dane z komputera zdalnego możesz skopiować na komputer lokalny za pomocą polecenia `scp`. Będąc cały czas zalogowanym na serwer `eto` stwórz w katalogu domowym plik `copyme`. Wyloguj się poleceniem `exit` lub skrótem klawiaturowym `Ctrl+D`.

Otwórz konsolę na komputerze lokalnym i ściągnij utworzony plik za pomocą polecenia

```
scp stud01@info3.meil.pw.edu.pl:~/copyme ./
```

W ogólności składnia ma postać

```
scp login@host:sciezka_do_pliku gdzie_zapisac
```

Program `scp` działa tak jak `cp`, z tą różnicą, że cel (lub źródło) znajduje się na innym komputerze obsługującym połączenia ssh. Dla komputerów z systemem z rodziny Windows stworzono program `WinSCP`, który pozwala na kopiowanie danych w trybie graficznym.

Deser! *

Pre-rekwizyty

Sprawdź do czego służy program `write` z użyciem polecenia `man`. Porównaj także rezultaty komend:

```
$ who
$ who | awk '{print $1}'
$ who | awk '{print $2}'
```

Zwróć uwagę na symbol |, który oznacza przekierowanie wyjścia jednego programu na wejście drugiego. Taki zapis nazywa się "potokiem" (ang. pipe).

Skrypt spamera

Stwórz skrypt:

```
#!/bin/bash

for u in $( who | awk '{print $1}' )
do
   echo $u
done
```

i sprawdź jak działa. Następnie zmodyfikuj go tak, aby "zaspamować" wszystkich zalogowanych.