Polecenia powłoki

Powłoka systemowa (ang. *shell*) to program będący pośrednikiem pomiędzy systemem operacyjnym a użytkownikiem. Można powiedzieć, że pełni funkcję interpretera kolejno wpisywanych poleceń. W systemach uniksowych najpopularniejszą powłoką jest *bash* (ang. *Bourne-Again SHell*).

Podczas tego kursu będziemy pracować w systemie *Linux* (polecenie `lsb_release -a` wyświetla informacje jakiej dystrybucji używamy). W wierszu poleceń (*terminal*) wpisujemy `ssh nazwa_użytkownika@student.tcs.uj.edu.pl` aby zalogować się na konto użytkownika na serwerze dla studentów. W systemie *Windows* możemy wykonać tę operację za pomocą programu *putty* (należy go pobrać z internetu).

Polecenie `ls -al` wyświetla całą zawartość bieżącego katalogu wraz z dodatkowymi informacjami na temat plików i folderów. Przykładowy wynik wykonania polecenia `ls -al`:

```
total 12
drwxrwxr-x  3 kecim kecim 4096 Sep 22 10:47 .
drwx-----x 44 kecim users 4096 Sep 21 22:09 ..
lrwxrwxrwx  1 kecim kecim   22 Sep 22 10:47 myD -> /home/kecim/Downloads/
-r--r-----  1 kecim kecim  100 Sep 22 10:40 plik-admina
-rw-rw-r--  1 kecim kecim   32 Sep 22 10:39 README.txt
drwxrwxr-x  2 kecim kecim 4096 Sep 21 22:17 Srodowisko
```

W kolejnych kolumnach wyświetlane są następujące informacje:

1. Pierwszy znak odpowiada za typ pliku: `-` to zwykły plik, `d` katalog, `l` link (skrót) do pliku, `b` albo `c` oznaczają urządzenie np. dysk twardy. Pozostałe 9 znaków odpowiada za prawa dostępu do pliku/katalogu/zasobu. Pierwsze 3 odpowiadają za prawa aktualnego użytkownika, następne 3 za prawa grupy, ostatnie 3 za prawa wszystkich użytkowników. Istnieją 3 podstawowe prawa, kolejno: `r` (read) prawo odczytu, `w` (write) prawo zapisu, `x` (execute) prawo wykonania, natomiast `-` oznacza, że użytkownik nie ma danego prawa.

2. Liczba w kolumnie oznacza ile bloków pamięci zajmuje dany plik/katalog (dla pliku jest to 1, dla folderu ilość folderów znajdujących się bezpośrednio w nim)

3. Właściciel pliku/katalogu.

4. Grupa do której należy właściciel pliku/katalogu.

5. Rozmiar pliku/katalogu w bajtach (rozmiar katalogu jest zawsze taki sam, nie są w niego wliczone rozmiary plików i katalogów które zawiera).

6-8. Data utworzenia/zmodyfikowania pliku/katalogu.

9. Nazwa pliku/katalogu.

Plik/katalog jest ukryty gdy jego nazwa zaczyna się od kropki. Zwróć uwagę na dwa pierwsze katalogi w przykładzie „.” i „..” reprezentują one odpowiednio aktualny katalog i

katalog nadrzędny. Takie dwa ukryte katalogi znajdują się w każdym katalogu. Szczególny przypadek to katalog główny „/”, który sam dla siebie jest katalogiem nadrzędnym, a więc „.” i „..” reprezentują tam ten sam katalog.

Podstawowe polecenia powłoki (nie tylko basha):

`passwd`	zmienia hasło użytkownika
`whoami`	wyświetla nazwę aktualnego użytkownika
`who`	wyświetla wszystkich użytkowników zalogowanych aktualnie w systemie
`pwd`	podaje aktualny katalog wraz ze ścieżką dostępu od katalogu głównego
`clear`	czyści ekran terminalu
`exit`	zamyka powłokę
`ls`	wyświetla zawartość podanego katalogu
`cd`	zmienia aktualny katalog na podany
`man`	wyświetla pomoc (dokumentację) dla podanego polecenia
`touch`	uaktualnienia czas ostatniej modyfikacji pliku (tworzy go, jeśli nie istniał)
`mkdir`	tworzy katalog o podanej nazwie
`rm`	usuwa pliki i katalogi (proponujemy nie używać `rmdir`)
`cp`	kopiuje pliki i katalogi
`mv`	przenosi (zmienia nazwę) plików i katalogów
`chmod`	zmienia prawa własności plików i katalogów
`echo`	wyświetla podaną linię tekstu
`cat`	łączy pliki i wypisuje je na standardowe wyjście
`tac`	to samo co `cat`, ale linie w odwrotnej kolejności
`wc`	liczy linie, słowa i znaki danego pliku
`head`	wyświetla początkowe linie pliku
`tail`	wyświetla końcowe linie pliku
`sort`	sortuje linie
`uniq`	omija powtarzające się linie
`quota`	wyświetla wykorzystanie miejsca na koncie użytkownika i jego limit
`du`	wyświetla rozmiar pliku albo katalogu
`find`	szuka plików w podanym katalogu

Przykłady użycia:

`ls`	wyświetlenie zawartości aktualnego katalogu
`ls Polska`	wyświetlenie zawartości katalogu „Polska”
`ls -l`	wyświetlenie zawartości aktualnego katalogu z dodatkowymi informacjami o plikach i katalogach (np. atrybuty, przynależność)
`ls -a`	wyświetlenie zawartości aktualnego katalogu wraz z plikami ukrytymi
`ls -la Polska`	wyświetlenie zawartości katalogu „Polska” wraz z plikami ukrytymi i z dodatkowymi informacjami o plikach
`ls -R`	wyświetlenie zawartości aktualnego katalogu i jego podkatalogów

`cd`	zmiana katalogu na katalog domowy użytkownika
`cd Polska`	zmiana katalogu na „Polska”
`cd ..`	zmiana katalogu na nadrzędny względem aktualnego
`cd .`	zmiana katalogu na aktualny katalog (nic się nie dzieje)
`cd /`	zmiana katalogu na katalog główny
`cd ~`	zmiana katalogu na katalog domowy użytkownika (to samo, co polecenie `cd` bez argumentów)
`cd ../Polska/Warszawa`	przykład zmiany katalogu (ścieżka względna) na katalog „Warszawa” w katalogu „Polska’, który znajduje się w katalogu nadrzędnym względem aktualnego katalogu
`cd /home`	przykład zmiany katalogu (ścieżka bezwzględna) na katalog „home” w katalogu głównym
`man cd`	pomoc dla polecenia `cd`
`man man`	pomoc dla polecenia `man`
`touch a b`	tworzenie (aktualizacja daty modyfikacji) plików „a” i „b”
`touch a\ b`	tworzenie (aktualizacja daty modyfikacji) pliku „a b”
`touch "a b"`	tworzenie (aktualizacja daty modyfikacji) pliku „a b”
`mkdir Niemcy`	utworzenie katalogu „Niemcy”
`mkdir Niemcy/Berlin`	utworzenie katalogu „Berlin” w istniejącym katalogu „Niemcy”
`mkdir -p Grecja/Kos`	utworzenie katalogu „Kos” w NIEKONIECZNIE istniejącym katalogu „Grecja” (może zostać utworzony)
`rm a b`	usuwanie plików „a” i „b”
`rm *`	usuwanie wszystkich plików z aktualnego katalogu poza plikami ukrytymi
`rm * .*`	usuwanie wszystkich plików (także ukrytych) z aktualnego katalogu
`rm *test*`	usuwanie wszystkich nieukrytych plików, które posiadają słowo „test” w nazwie
`rm m[aiu]d`	usuwanie plików „mad”, „mid”, „mud”, o ile te istnieją
`rm *.txt`	usuwanie nieukrytych plików o nazwach kończących się na „.txt”
`rm -r Niemcy`	usuwanie katalogu „Niemcy” z całą zawartością
`rm -r Polska/Krakow`	usuwanie podkatalogu „Krakow” z katalogu „Polska”
`cp x y`	kopiowanie zawartości pliku „x” do pliku „y”
`cp x y Polska`	kopiowanie plików „x” i „y” do katalogu „Polska”
`cp c?t Polska`	kopiowanie plików trzyliterowych o pierwszej literze „c”, drugiej dowolnej i trzeciej „t” do katalogu „Polska”
`cp * Francja`	kopiowanie wszystkich nieukrytych plików (ale nie nieukrytych katalogów!) do katalogu „Francja”
`cp -r Polska USA`	jeżeli katalog „USA” istnieje, to kopiowanie katalogu „Polska” do katalogu „USA”; w.p.p. utworzenie katalogu „USA” o takiej samej zawartości jak katalog „Polska”

`mv x y`	zmiana nazwy pliku z „x” na „y”
`mv x y Polska`	przeniesienie plików „x” i „y” do katalogu „Polska”
`mv a[^cr]t Polska`	przeniesienie plików trzyliterowych o pierwszej literze „a”, drugiej innej niż „c”, „r” i trzeciej „t” do katalogu „Polska”
`mv fr* Francja`	przeniesienie wszystkich plików i katalogów o nazwach zaczynających się od „fr” do katalogu „Francja”
`mv Polska USA`	jeżeli katalog „USA” istnieje, to przeniesienie katalogu „Polska” do katalogu „USA”; w.p.p. zmiana nazwy katalogu „Polska” na „USA”
`chmod 764 p1`	zmiana praw własności do pliku „p1” na: właściciel - prawo do czytania(4), zapisu(2) i wykonania(1), grupa - tylko prawo do czytania(4) i zapisu(2), inni - tylko prawo do czytania(4)
`chmod 764 Kat -R`	zmiana praw własności do katalogu „Kat” i całej jego zawartości na podane wyżej
`chmod a+r p2`	nadanie praw wszystkim do czytania pliku „p2”
`chmod +r p2`	również nadanie praw wszystkim do czytania pliku „p2”
`chmod u+w p2`	nadanie praw użytkownikowi do zapisu pliku „p2”
`chmod g-w p2`	usunięcie praw grupie do zapisu pliku „p2”
`chmod o-x p2`	usunięcie praw innym do wykonania pliku „p2”
`chmod u=rw p2`	zmiana praw użytkownika do pliku „p2” na: czytanie, zapis, brak możliwości wykonania
`chmod go=x p2`	zmiana praw grupy i innych do pliku „p2” na: brak możliwości czytania, brak możliwości zapisu, wykonanie
`echo "Hej"`	wyświetlanie `Hej` na standardowym wyjściu
`echo "Hej\nHej"`	wyświetlanie `Hej\nHej` na standardowym wyjściu
`echo -e "Hej\nHej"`	wyświetlanie 2 linijek `Hej` na standardowym wyjściu
`cat`	wczytywanie ze standardowego wejścia kolejnych linijek tekstu i wypisywanie ich na standardowe wyjście (Ctrl-D kończy proces)
`cat input`	wypisywanie pliku „input” na standardowe wyjście
`cat < input`	również wypisywanie pliku „input” na standardowe wyjście
`cat > output`	wczytywanie ze standardowego wejścia kolejnych linijek tekstu i zapisywanie ich do pliku „output”
`cat >> output`	wczytywanie ze standardowego wejścia kolejnych linijek tekstu i dopisywanie ich na koniec pliku „output”
`cat in1 in2 > out`	stworzenie (nadpisanie) pliku „out” poprzez połączenie zawartości plików (kolejno) „in1” i „in2”
`cat -n input`	wypisywanie pliku „input” wraz z numerami linii
`cat -E input`	wypisywanie pliku „input” wraz z końcami linii (symbol `$`)
`tac`	to samo co polecenie `cat`, ale linijki na standardowym wyjściu wypisywane są w odwrotnej kolejności
`tac input`	wypisywanie pliku „input” na standardowe wyjście od ostatniej linijki do pierwszej

`wc p1`	zliczanie (kolejno) linijek, słów i znaków pliku „p1”
`wc -l p1`	zliczanie linijek pliku „p1”
`wc -w p1`	zliczanie słów pliku „p1”
`wc -c p1`	zliczanie znaków pliku „p1”
`head text`	wypisywanie pierwszych 10 linijek pliku „text”
`head -5 text`	wypisywanie pierwszych 5 linijek pliku „text”
`tail text`	wypisywanie ostatnich 10 linijek pliku „text”
`tail -3 text`	wypisywanie ostatnich 3 linijek pliku „text”
`sort`	wczytywanie ze standardowego wejścia kolejnych linijek tekstu i (po Ctrl-D) wypisywanie ich na standardowe wyjście w postaci posortowanej leksykograficznie
`sort < plik1`	wypisywanie posortowanych linijek pliku „plik1”
`sort -r < plik1`	wypisywanie odwrotnie posortowanych linijek pliku „plik1”
`uniq`	wczytywanie ze standardowego wejścia kolejnych linii tekstu i wypisywanie ich na standardowe wyjście, o ile kolejna wpisana linia nie jest taka sama jak poprzednia (Ctrl-D kończy proces)
`uniq < plik2`	wypisywanie linijek pliku „plik2” z ominięciem powtarzających się po sobie identycznych linijek
`du p1`	wyświetla rozmiar pliku „p1”, domyślnie ilość zajmowanych bloków pamięci
`du -b tmp`	wyświetla w bajtach rozmiar katalogu „tmp” oraz rekurencyjnie jego podkatalogów
`find ~ p1`	szuka pliku „p1” w katalogu domowym użytkownika
`find /bin p2`	szuka pliku „p2” w katalogu „bin” z katalogu głównego

Jak widać, w przypadku poleceń czytających ze standardowego wejścia możliwe jest *przekierowanie* do nich zawartości danego pliku przy pomocy symbolu `<`. Jeżeli chcemy, aby efekt działań danego polecenia zamiast na standardowe wyjście został przekierowany do wybranego przez nas pliku, używamy symboli `>` lub `>>` (patrz: polecenie `cat`). Można też zrobić tak: `> p1`. Jeśli plik „p1” nie istnieje, to zostanie utworzony; w przeciwnym przypadku jego zawartość zostanie wyzerowana.

Jeżeli naszym celem jest potraktowanie wyjścia jednego polecenia jako wejścia do kolejnego, to możemy skorzystać z *potoków* (symbol: `|`). Przykładowo, jeśli chcemy posortować odwrotnie leksykograficznie pierwszych 25 linijek pliku „file1”, to możemy wykorzystać następującego *jednolinijkowca* (ang. *oneliner*):

```
head -25 file1 | sort -r
```

Oczywiście, nie ma przeszkód, aby wynik polecenia wykorzystującego potoki zapisać do pliku, np. „file2”:

```
head -25 file1 | sort -r >file2
```

Poprzez sprytne wykorzystanie potoków możemy poradzić sobie z tak naturalnymi pro-

blemami, jak zliczenie niepowtarzających się linii pliku „p2”:

```
sort <p2 | uniq | wc -l
```

```
cat p2 | sort | uniq | wc -l
```

Quota czyli przydzielona użytkownikowi ilość miejsca na przechowywanie danych może zostać przekroczona. Nie można wtedy uruchomić trybu graficznego na wydziałowym koncie użytkownika. Aby rozwiązać ten problem należy usunąć pliki/foldery które zajmują zbyt dużo miejsca. Żeby znaleźć 10 największych plików lub folderów znajdujących się na koncie użytkownika należy użyć komendy:

```
du -a ~ | sort -n -r | head -n 10
```

Aby znaleźć wszystkie pliki większe niż 20 megabajtów znajdujących się na koncie użytkownika należy użyć polecenia:

```
find ~ -size +20M
```

Podstawy pisania skryptów w Bashu

Najprostszy skrypt w bashu to ciąg kolejnych poleceń powłoki zapisany w pliku tekstowym. Znak `#` oznacza komentarz obejmujący swoim zasięgiem przestrzeń od tego znaku do końca linii. Natomiast `!#` jest znakiem wskazującym na interpreter, który zostanie użyty do wykonania skryptu, w naszym przypadku będzie to `/bin/bash`. Linia `#!/bin/bash` powinna być umieszczona na początku każdego skryptu. Przykłady:

```
#!/bin/bash
echo "Witaj shell-u!"
#skrypt wypisujący na standardowe wyjście tekst:
# Witaj shell-u!
```

Każde kolejne polecenie w skrypcie powinno być wpisana w nowej linii lub oddzielone od poprzedniego średnikiem:

```
#!/bin/bash
echo "Witaj shell-u!"; echo "Witajcie studenci!"
echo "Koncze prace."
#skrypt wypisujący trzy linie tekstu na standardowe wyjscie:
# Witaj shell-u!
# Witajcie studenci!
# Koncze prace.
```

Plik ze skryptem powinien mieć rozszerzenie `.sh`, np. `helloshell.sh`. Aby wykonać skrypt musimy nadać mu odpowiednie prawa dostępu, a następnie go uruchomić:

```
chmod 711 helloshell.sh
./helloshell.sh
```

Podczas uruchomienia skryptu możemy podać mu argumenty:

```
./przyjmuje-argumenty.sh pierwszy drugi trzeci
```

Aby wewnątrz skryptu dostać się do podanych argumentów używamy zmiennych `$1`, `$2`, itd. Dokładniej ich użycie zostanie omówione później, teraz zademonstrujemy prosty przykład:

```
#!/bin/bash
#to jest plik: przyjmuje-argumenty.sh
echo "Witaj $1!"
echo "Witaj $2!"
echo "Witaj $3!"
echo "Zegnajcie $1, $2 i $3"
```

Wykonanie tego skryptu, z argumentami podanymi tak jak powyżej, zwróci nam następujący wynik:

```
Witaj pierwszy!
Witaj drugi!
Witaj trzeci!
Zegnajcie pierwszy, drugi i trzeci!
```

Uruchomienie skryptu `./kreator.sh Skrypty skrypt.sh` utworzy nam katalog „Skrypty" a w jego wnętrzu plik "skrypt.sh". Zwróć uwagę na to jaki będzie aktualny folder po wykonaniu skryptu.

```
#!/bin/bash
#to jest plik: kreator.sh
mkdir $1
cd $1
touch $2
```

Aktualny katalog nie zmienia się pomimo umieszczenia w skrypcie polecenia `cd $1`. Dzieje się tak ponieważ skrypty są uruchamiane w *podpowłoce*, jako że *powłoka systemowa* jest programem może uruchamiać sama siebie, a każda *podpowłoka* ma ustawiony własny aktualny katalog. Po wykonaniu skryptu *podpowłoka* jest zamykana. Temat ten zostanie szerzej omówiony później. Natomiast aby wywołać skrypt w bieżącej powłoce należy wpisać `. nazwa_skryptu` zamiast `./nazwa_skryptu`. Domyślnie będziemy uruchamiać skrypty w *podpowłoce*.