Polecenia powłoki 2

`less` to program do przeglądania plików tekstowych w konsoli. **Uwaga:** `less` nie jest edytorem. Jedną z jego zalet jest to, że nie wczytuje on całego pliku do pamięci przed wyświetleniem, dzięki czemu działa na dużych plikach szybciej od standardowych edytorów. Standardowe wywołanie programu to:

```
less nazwa_pliku
```

Oto lista niektórych komend użytecznych przy nawigacji:

↑ lub ↓	przesunięcie o jedną linię (`j` i `k` dają ten sam efekt),
`g`	przejście do początku pliku
`G`	przejście do końca pliku
`/wzorzec`	wyszukanie wzorca od bieżącej linii do przodu,
`n`	wyszukanie następnego wystąpienia wzorca,
`N`	wyszukanie poprzedniego wystąpienia wzorca,
`F`	przejście do trybu oczekiwania na dopisywanie linii do pliku,
`q`	wyjście z programu.

A oto kilka użytecznych wywołań `less`:

```
less +150 filename
less +G filename
less +F filename
```

`tr` to program do zamiany lub usuwania znaków. Składnia jego wywołania jest następująca: `tr [opcje] "set1" "set2"`, gdzie `set1` jest zbiorem znaków, które chcemy zmienić w odpowiadające im znaki w zbiorze `set2`. Jeżeli w użyciu jest opcja `-d` (usuwanie znaków), to `"set2"` nie występuje. Przykładowe wywołania:

```
echo "abczdefcba azaaz" | tr "abc" "def"
⟶defzdeffed dzddz
echo "abczdefcba azaaz" | tr "abcz" "def"
⟶deffdeffed dfddf
echo "abczdefcba azaaz" | tr -d "abc"
⟶zdef zz
```

Zamiast wypisywać kolejne znaki, możemy zastosować notację zbiorów:

```
echo 'Linux' | tr "a-z" "A-Z"
echo 'Linux' | tr "[:lower:]" "[:upper:]"
```

Opcja `-c` sprawia, że polecenie `tr` zamiast zbioru `set1` przyjmuje jego dopełnienie. Opcja `-s` powoduje zamianę **ciągu** znaków ze zbioru `set1`, na **jeden** odpowiadający mu znak ze zbioru `set2`.

```
tr -cs "[:alpha:]" "\n" < input.txt
tr -cd "[:print:]" < input.txt
```

`tr -s ' '` (zamienia wielokrotne, kolejne wystąpienia znaku ze zbioru na pojedynczy znak; zatem powyższe wywołanie)

`diff` to program do porównywania zawartości dwóch plików. Wypisuje listę modyfikacji (`a` – dodanie linii, `c` – zamiana linii, `d` – usunięcie linii), które trzeba zastosować do linii w pliku pierwszym, aby uzyskać plik drugi. Standardowe wywołanie:
`diff plik1 plik2`
Oto lista niektórych przydatnych opcji dla polecenia `diff`:

`-b`	ignoruje zmiany w liczbie białych znaków,
`-B`	ignoruje puste linie,
`-i`	ignoruje wielkość liter,
`-I regexp`	ignoruje linie pasujące do wyrażenia regularnego `regexp`,
`-q`	podaje tylko informację czy pliki się różnią,
`-r`	rekurencyjnie porównuje zawartość dwóch katalogów,
`-u`	zmienia format raportu porównania na zwarty kontekstowy,
`-w`	ignoruje białe znaki.

`aspell` to program do sprawdzania pisowni, a przede wszystkim wychwytywania literówek. Program ma filtry dopasowane do plików `html`, `tex/latex` i kilku innych. Do sprawdzania pisowni można podłączyć zewnętrzne słowniki. Poprawianie błędów jest wykonywane poprzez interaktywną formułę: `aspell` wskazuje błąd oraz podobne słowa w słowniku, użytkownik decyduje czy (i co) robić ze wskazanym słowem.

Przykłady użycia:
```
aspell check file.txt
aspell check --lang=pl file.txt
```

Poprzez komendę `pipe` program `aspell` wskazuje literówki bez interakcji, co jest bardziej wskazane, jeśli chcemy dokonać analizy statystycznej naszych błędów. Zatem polecenie `aspell pipe < file.txt` wydrukuje na standardowe wyjście jedną linię dla każdego słowa w pliku `file.txt`. Jeśli słowo jest w słowniku linia będzie zawierać tylko jeden znak `*`. Jeśli słowo nie jest w słowniku, ale `aspell` ma sugestie co to powinno być, wtedy linia wygląda tak:
`& original count offset: miss, miss, ...`
Jeśli słowo nie jest w słowniku i `aspell` nie ma sugestii, wtedy linia wygląda tak:
`# original offset`

`ssh` to program umożliwiający połączenie z serwerem przy pomocy protokołu *secure shell*. Protokół ten jest następcą protokołu *Telnet* i służy do terminalowego łączenia się ze zdalnym komputerami. Obecnie *SSH* to wspólna nazwa dla całej większej rodziny protokołów (*SCP*, *SFTP*). Wspólną cechą tych protokołów jest technika szyfrowania i rozpoznawania użytkownika.

Podstawowe użycie programu `ssh` wygląda tak:
`ssh username@student.tcs.uj.edu.pl`

Przy pierwszej próbie połączenia z danym serwerem `ssh` poda jak się przedstawia znale-

ziony zdalny komputer i czy na pewno chcemy się połączyć oraz dołożyć jego i jego klucz RSA do listy znanych hostów (`~/.ssh/known_hosts`).

`ssh` umożliwia również uwierzytelnianie poprzez klucz publiczny i prywatny. Jest to bezpieczniejszy i często wygodniejszy sposób w praktyce. Aby wygenerować klucze piszemy:
`ssh-keygen -t rsa`
Klucz generowany jest z tzw. *passphrase* (którym nie powinniśmy się dzielić), z założenia powinna to być dłuższa sekwencja znaków od standardowego hasła. Najczęściej jest to zdanie w języku naturalnym. Klucz prywatny i publiczny domyślnie zapisują się w katalogu `~/.ssh/` w plikach `coś` i `coś.pub`, odpowiednio. Następnie powinniśmy skopiować klucz publiczny do odpowiedniego katalogu na zdalnym komputerze:
`scp ~/.ssh/coś.pub username@student.tcs.uj.edu.pl:.ssh/authorized_keys`

X Window System to graficzny system okien odpowiedzialny za wyświtlanie interfejsu użytkownika, dostępny w wielu systemach Linux. Jedną z mniej znanych zalet *X Window* jest ich niesamowita transparentność w sieci. Ten system okienkowy z założenia miał być gotowy transmisji *GUI* i bitmap. W skrócie jeśli mamy na lokalnym i zdalnym komputerze *X Window* to możemy połączyć się przez `ssh` i odpalić zdalnie np. przeglądarkę internetową. Połączenie tunelujące protokół *X11* włączamy opcją `-X`:
`ssh -X username@student.tcs.uj.edu.pl`

Można też nawiązywać połączenie ze zdalnym komputerem celem wykonania jednego polecenia w terminalu. Jest to szczególnie użyteczne, gdy jesteśmy na maszynie A, chcemy pracować na maszynie C i jedyny sposób połączenia się z tą maszyna wiedzie poprzez maszynę B.
`ssh username@student.tcs.uj.edu.pl ls -al`
`ssh username@B ssh username@C`

`scp` to program, który używa protokołu *SSH* do przesyłania plików pomiędzy lokalnym i zdalnym komputerem.
`scp filename username@student.tcs.uj.edu.pl:`
`scp filename username@student.tcs.uj.edu.pl:directory/new_filename`
`scp username@student.tcs.uj.edu.pl:filename newfilename`
`scp -r directory username@student.tcs.uj.edu.pl:`

`wget` to prosty program do pobierania plików za pośrednictwem protokołu *HTTP* lub *FTP*.
`wget http://srodowisko.tcs.uj.edu.pl/docs/srodowisko-2015-sed.pdf`
`wget -O taglist.zip http://.../download_script.php?id=1234`
`wget -c http://...`
`wget -b http://...`
`tail -f wget-log`
`wget -r http://...`

Polecenie `history` wyświetla historię wydanych poleceń w bashu. Może ona zawierać bardzo dużo wpisów, zatem najlepiej wywoływać ją np. w postaci `history | less` lub `history n`, gdzie `n` jest liczbą ostatnich poleceń (czyli `history 30` wypisze 30 ostatnio używanych komend). Do zawartości historii możemy odnosić się następująco:

`!!`	ostatnio używane polecenie,
`!5`	piąte polecenie z historii,
`!-3`	trzecie od końca polecenie,
`!l`	ostatnie polecenie zaczynające się na "l",
`!*`	lista argumentów (poza zerowym) ostatniego polecenia,
`!l:1`	pierwszy argument ostatniego polecenia zaczynającego się na "l",
`!!:1-3`	argumenty od 1 do 3 ostatniego polecenia.

Przez "argumenty" należy rozumieć kolejne wyrazy. Na przykład w poleceniu:
`cat -nE plik`
argumentem 0 jest `cat`, argumentem 1 – `-nE`, a argumentem 2 – `plik`. Warto zauważyć, że polecenia są zapamiętywane w historii już z podstawionymi konkretnymi instrukcjami (argumentami) w miejscu tych zaczynających się od `!`.

Historia wywołanych komend basha znajduje się domyślnie w pliku `~/bash_history`. Zawartość tego pliku może być jednak mniej aktualna od informacji z polecenia `history`, gdyż aktualizowana jest standardowo przy zamykaniu powłoki. Aby natychmiast zaktualizować plik `~/bash_history` należy użyć komendy: `history -a`

Przy pomocy polecenia `alias` można definiować nowe polecenia na bazie już istniejących. Samo polecenie `alias` bez argumentów wyświetla wszystkie komendy powłoki, które przy jego pomocy powstały. Przykładowo, jeżeli często używamy polecenia `ls -al`, to możemy je zastąpić jednym poleceniem `lsa` w następujący sposób:
`alias lsa='ls -al'`

Od tej pory, aż do zamknięcia powłoki, będziemy mogli używać samego polecenia `lsa` do dokładnego wypisywania zawartości aktualnego katalogu czy `lsa bin` do podobnego wyświetlania katalogu `bin`. W celu utrzymania tej komendy na stałe (czyli po zamknięciu i ponownym uruchomieniu basha) powinniśmy jej powyższą definicję wpisać do osobnej linijki w pliku `~/.bashrc`. W nim znajdują się zdefiniowane przez użytkownika funkcje, aliasy i zmienne, które nabierają znaczenia przy starcie basha. A zrestartować powłokę możemy przy pomocy polecenia:
`exec bash`