INFORMATYKA I: INSTRUKCJA 2 I 3

1 BASH: skrypty

Pisanie skryptów, polega na spisaniu w pliku komend, które normalnie wpisywalibyśmy w linii poleceń. Taki plik możemy następnie oznaczyć jako wykonywalny komendą `chmod +x plik` i wykonać komendą `./plik`. Linia poleceń (BASH) służy do uruchomiania programów — dlatego:
każda linijka skryptu wygląda nastepująco: „`program agumenty`”.
Przeanalizuj fragment kodu, z zaznaczonymi programami i opcjami:

```
i=1
while test $i -lt 10
do
echo $i
cp plik plik_$i
i=$(expr $i + 1)
done
```

Gdy zapamiętamy tą zasadę, łatwo zobaczyć, że:

- `i=1` piszemy bez spacji ponieważ wtedy BASH wie, że to przypisanie, a nie program `i` z opcjami `=` i `1`.
- w wyrażeniu `expr $i + 1`, musimy zachować spacje, żeby program `expr` dostał trzy argumenty „`$i`”, „`+`” i „`1`”, a nie jeden „`i+1`”.
- w pętli `while`, nie możemy wpisać „`i<10`”, lecz musimy użyć jakiegoś programu. Do wszelkiego rodzaju testów stwożony został program `test`. W tym wypadku podajemy mu za argumenty „`$i`”, „`-lt`” i „`10`”, gdzie opcja `-lt` oznacza „less than”.

1.1 Przydatne programy

Jeśli już wiemy, że każdy skrypt w BASH to seria wywołanych programów, to potrzebne jest nam dużo małych programów, z których będziemy mogli tworzyć skrypty.

- `echo tekst` — Wpisuje tekst na ekran.
- `cat plik` — Wypisuje zawartość pliku na ekran
- `grep tekst` — Czyta z klawiatury tekst i wypisuje tylko linie zawierające tekst
- `grep tekst pliki` — Wyszukuje tekst w plikach
- `cd katalog` — Wchodzi do katalogu
- `ls katalog` — Wypisuje zawartość katalogu na ekran
- `cp pliki katalog` — Kopiuje pliki do katalogu
- `cp plik1 plik2` — Kopiuje plik o nazwie plik1 do pliku o nazwie plik2
- `mv pliki katalog` — Przenosi pliki do katalogu
- `mv plik1 plik2` — Zmienia nazwę pliku z plik1 na plik2
- `sed 's/tekst1/tekst2/g'` — Czyta z klawiatury tekst i go wypisuje zamieniając „tekst1” na „tekst2”

1.2 Przekierowanie wejścia wyjścia

Standardowo wszystkie programy czytają z klawiatury i piszą na ekran. Można jednak zarówno pierwsze jak i drugie przekierować.

- `program > plik` — To co program wypisałby na ekran, zostanie wpisane do pliku (plik zostanie nadpisany jeśli istnieje)
- `program >> plik` — To co program wypisałby na ekran, zostanie dopisane do pliku (plik zostanie utwożony jeśli nie istniał)
- `program < plik` — Program dostanie zawartość pliku, tak jakbyśmy ją wpisali z klawiatury
- `program1 | program2` — To co program1 wypisałby na ekran, zostanie wpisane „z klawiatury” do program2
- `` `program` `` lub `$(program)` — To co program wypisałby na ekran, zostanie wklejone w tym miejscu kodu (patrz przykłady). Znak ` jest na klawiaturze przy tyldzie ~.

Przykłady:

- `echo Tekst > plik` — wypisze „Tekst” do pliku (plik zostanie nadpisany jeśli istnieje)

- `echo Tekst >> plik` — dopisze „Tekst” do pliku (plik zostanie utworzony jeśli nie istniał)
- `grep Tekst < plik` — wyszuka w pliku linie zwierające „Tekst” i je wypisze na ekran
- `echo Tekst | sed 's/st/a/g'` — Zamieni w „Tekst” każde wystąpienie „st” na „a”. Więc wypisze na ekran „Teka”.
- `echo $nazwa | sed 's/\.txt/.dat/g'` — Zastąpi w zmiennej `nazwa` końcówkę `.txt` na `.dat`. Rezultat wypisze na ekran.
- `echo $nazwa | sed 's/\.txt/.dat/g'` — Zastąpi w zmiennej `nazwa` końcówkę `.txt` na `.dat`. Rezultat wypisze na ekran.
- `nazwa2=$(echo $nazwa | sed 's/\.txt/.dat/g')` — Jak poprzednio, lecz rezultat wypisze do zmiennej `nazwa2`.
- `ls katalog > plik` — wypisze zawartość katalogu do pliku (plik zostanie nadpisany jeśli istnieje)
- `cp ‘ls‘ katalog` albo `cp $(ls) katalog` — skopiuje pliki do `katalogu` według listy zwróconej przez `ls`.
- `cp ‘cat plik‘ katalog` bądz `cp $(cat plik) katalog` — skopiuje pliki do `katalogu` według listy zawartej w `pliku`.

1.3 Pętle i wyrażenia warunkowe

- ```
  if program argumenty
  then
  polecenia1
  else
  polecenia2
  fi
  ```
  Jeśli wykonanie „program argumenty” się powiedzie (program zwróci 0), to wykonane zostaną polecenia1. W przeciwnym wypadku wykonane zostaną polecenia2.
- ```
  while program argumenty
  do
  polecenia
  done
  ```
  Pętla, która będzie wykonywać polecenia, puki „program argumenty” będzie wykonywany z powodzeniem.
- ```
  for i in lista
  do
  polecenia
  done
  ```
  Pętla, która po kolei każdy element listy wstawi do zmiennej `i`, a następnie wykona polecenia.

Dla przykładu:
```
for i in *.jpg
do
mv $i IMG/a_$i
done
```
Przeniesie każdy plik o końcówce `.jpg`, do katalogu `IMG` dodając im przedrostek `a_` (np.: `obrazek.jpg` zamieni na `IMG/a_obrazek.jpg`).

1.4 Ćwiczenia

Domyślnym edytorem na serwerze `info3` jest edytor `nano` , dostępny jest teś edytor `vim` . Pierwszy z nich wydaje się prostszy w obsłudze, drugi występuje na prawie każdym komputerze z UNIXem.

- Przy pomocy pętli wypisz na ekran liczby od 0 do 10
- Zmień skrypt, tak aby wypisywał od 0 do podanej jako argument wielkości

2 Obróbka obrazków

2.1 `convert`

Głównym programem którego będziemy używać to `convert` z biblioteki ImageMagick. Program ten służy do najróżniejszego typu konwersji i zmiany właściwości obrazów — lecz potrafi także dodawać elementy do obrazu, a nawet tworzyć obrazy od zera. Najłatwiej zobaczyć jego użycie na przykładach:
UWAGA: Zanim zaczniesz, skopiuj katalog ze zdjęciami do jakiegoś tymczasowego katalogu!

- `convert plik.gif plik.jpg` — przekonwertuje plik w formacie GIF na format JPEG
```

- `convert plik1.jpg -resize 50% plik2.jpg` — zmniejszy obrazek dwukrotnie
- `convert plik1.jpg -resize 100 plik2.jpg` — zmniejszy obrazek, tak by krótszy wymiar był 100 pikseli
- `convert plik1.jpg -resize 100x100 plik2.jpg` — zmniejszy obrazek tak, by mieścił się w kwadracie 100 na 100 pikseli
- `convert plik1.jpg -resize 100x100\! plik2.jpg` — zmniejszy obrazek dokładnie do rozmiaru 100 na 100 pixeli
- `convert -size 320x85 canvas:none -font Bookman-DemiItalic -pointsize 72 -draw "text 25,60 'Magick'" -channel RGBA -blur 0x6 -fill darkred -stroke magenta -draw "text 20,55 'Magick'" fuzzy-magick.jpg` — stworzy obrazek fuzzy-magick.jpg, z tekstem "Magick"

Wykonaj powyższe operacja, sprawdź efekty.

Ćwiczenia

Napisz skrypt który:

- Zmniejszy wszystkie pliki `jpg`
- Napisz skrypt który: Zmniejszy wszystkie pliki `jpg` umieszczając je w innym katalogu
- Napisz skrypt który: Skonwertuje wszystkie pliki `jpg` na `gif`, dodając końcówkę: `plik.jpg` → `plik.jpg.gif`
- Napisz skrypt który: Skonwertuje wszystkie pliki `jpg` na `gif`, zamieniając końcówkę `plik.jpg` → `plik.gif`
- Na każde zdjęcie naniesie tekst używając `-pointsize rozmiar -draw "text x,y 'Tekst'"`
- Na każde zdjęcie naniesie aktualną datę (komenda `date`)
- Na każde zdjęcie naniesie datę utworzenia tego zdjęcia (można ją wyciągnąć przy pomocy `stat -c %y plik`)
- Zmniejszy wszystkie obrazki z katalogu `drop1` i połączy je w animację przy pomocy `convert *.jpg animacja.gif`