



Informatyka III: Instrukcja 2

Skrypty BASH

1 Materiały

Materiały do tego laboratorium można znaleźć w katalogu `~laniewski/lab2_gorne` lub `~laniewski/lab2_dolne` na `info3.meil.pw.edu.pl` (górne: sala 120, dolne: sala 20B).

Pod linuxem należy w terminalu wykonać:

```
z scp [login]@orange.meil.pw.edu.pl:~laniewski/lab2_gorne .
z ./lab2_gorne [login na info3.meil.pw.edu.pl]
```

Następnie wyczyść terminal i wyczyść nowy. Polecenia te zagraj przygotowaną wersję programu ImageMagick i zainstaluj, a także utwórz katalog LAB2 w katalogu domowym, w który zawartość przykładowe obrazki do obróbki.

2 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 uruchamiania programów — dlatego:

kada linijka skryptu wygląda następująco: „program argumenty”.

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 te zasady, 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 $i + 1`, musimy zachować spację, aby program `expr` dostał trzy argumenty „`$i`”, „`+`” i „`1`”, a nie jeden „`i+1`”.
- w pętli `while`, nie możemy wpisać „`i<10`”, lecz musimy użyć jakiego programu. Do wszelkiego rodzaju testów stworzony został program `test`. W tym wypadku podajemy mu za argumenty „`$i`”, „`-lt`” i „`10`”, gdzie opcja `-lt` oznacza „less than”.

2.1 Przydatne programy

Jeli już wiemy, że każdy skrypt w BASH to seria wywołanych programów, to potrzebne jest nam dużo różnych 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`”

2.2 Przekierowanie wejścia wyjścia

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

- `program z plik` — To co program wypisywałby na ekran, zostanie wpisane do pliku (`plik` zostanie nadpisany jeśli istnieje)



- `program && plik` — To co `program` wypisaby na ekran, zostanie dopisane do pliku (`plik` zostanie utworzony jeli nie istnia)
- `program i plik` — Program dostanie zawarto `pliku`, tak jakbyśmy j wpisali z klawiatury
- `program1 — program2` — To co `program1` wypisaby na ekran, zostanie wpisane „z klawiatury” do `program2`
- `‘program’` lub `$(program)` — To co `program` wypisaby na ekran, zostanie wklejone w tym miejscu kodu (patrz przykady). Znak ‘ jest na klawiaturze przy tyldzie ~.

Przykady:

- `echo Tekst & plik` — wypisze „Tekst” do `pliku` (`plik` zostanie nadpisany jeli istnieje)
- `echo Tekst && plik` — dopisze „Tekst” do `pliku` (`plik` zostanie utworzony jeli nie istnia)
- `grep Tekst i plik` — wyszuka w `pliku` linie zawierajce „Tekst” i je wypisze na ekran
- `echo Tekst — sed 's/st/a/g'` — Zamieni w „Tekst” kade wystpienie „st” na „a”. Wic wypisze na ekran „Teka”.
- `echo $nazwa — sed 's/\.txt/.dat/g'` — Zastpi w zmiennej nazwa kocwk .txt na .dat. Rezultat wypisze na ekran.
- `echo $nazwa — sed 's/\.txt/.dat/g'` — Zastpi w zmiennej nazwa kocwk .txt na .dat. Rezultat wypisze na ekran.
- `nazwa2=$(echo $nazwa — sed 's/\.txt/.dat/g')` — Jak poprzednio, lecz rezultat wypisze do zmiennej nazwa2.
- `ls katalog & plik` — wypisze zawarto `katalogu` do `pliku` (`plik` zostanie nadpisany jeli istnieje)
- `cp 'ls' katalog` albo `cp $(ls) katalog` — skopiuje pliki do katalogu wedug listy zwrconej przez `ls`.
- `cp 'cat plik' katalog` bdz `cp $(cat plik) katalog` — skopiuje pliki do katalogu wedug listy zawartej w pliku.

2.3 Ptle i wyraenia warunkowe

- if `program argumenty`
then
`polecenia1`
else
`polecenia2`
fi
Jeli wykonanie „`program argumenty`” si powiedzie (program zwrci 0), to wykonane zostan `polecenia1`. W przeciwnym wypadku wykonane zostan `polecenia2`.
- while `program argumenty`
do
`polecenia`
done
Ptla, ktra bdzie wykonywa `polecenia`, puki „`program argumenty`” bdzie wykonywany z powodzeniem.
- for i in `lista`
do
`polecenia`
done
Ptla, ktra po kolei kady element `listy` wstawi do zmiennej i, a nastpnie wykona `polecenia`.

Dla przykadu:

```
for i in *.jpg
do
mv $i IMG/a_.$i
done
```

Przeniesie kady plik o kocwce .jpg, do katalogu IMG dodajc im przedrostek a_ (np.: obrazek.jpg zamieni na IMG/a_obrazek.jpg).

2.4 wiczenia

Domylnym edytorem na serwerze info3 jest edytor nano , dostpny jest te edytor vim . Pierwszy z nich wydaje si prostszy w obsudze, drugi wystpuje na prawie kadm komputerze z UNIXem.

- Przy pomocy ptli wypisz na ekran liczby od 0 do 10



- Zmie skrypt, tak aby wypisywa od 0 do podanej jako argument wielkości

3 Obróbka obrazków

3.1 convert

Głównym programem którego będziemy używać to convert z biblioteki ImageMagick. Program ten służy do najprostszej 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 pikseli
- `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 operacje, sprawdź efekty.

wzrost

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 kółka: `plik.jpg → plik.jpg.gif`
- Napisz skrypt który: Skonwertuje wszystkie pliki jpg na gif, zamieniając kółka `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 położy je w animacji przy pomocy `convert *.jpg animacja.gif`