Obligatorisk innlevering 1 høsten 2014, INF3331

Nicolai Skogheim <nicolai.skogheim@gmail.com>

September 19, 2014

Oppgave 1.1

Løsning (essensen):

```
folder=$1
days=$2

find -$folder -type f -mtime -${days}d | xargs du -h | sort -h}
```

Find har flaggene `-type` og `-mtime`, for å velge hendholdsvis filer (`-type f`) og treff som har mtime innenfor de siste n dagene.

`xargs` redistribuerer argumentene som kommer fra find gjennom pipa, og sender dem til `du` som har `-h` (human readable) flagget for å vise størrelsen på formen 1kb i stedet for 1024.

Til sist sorteres alt via `sort`-kommandoen, som sorterer etter størrelse i human readable-format pga `-h`-flagget.

Utover det som er svaret på oppgaven har jeg gjort et par ting som ikke var krevd, men som er god skikk i bash-programmer som krever argumenter.

Det første er å sjekke at hvis det ikke er sendt med to argumenter (`$#` = antall argumenter), så skal det vises en liten melding med hjelp til brukeren.

Jeg valgte også legge input-parameterene i variabler bare for å få navn på dem.

Kjøring

```
14:36 $ bash list_new_files.sh file_tree 80
644K        file_tree/KqOWv/H/XhdhBbk
772K        file_tree/zg/Hu/vv/2KKnyIt5
780K        file_tree/KqOWv/MH/Z9kP8NB
780K        file_tree/Pkvye/vlfN/ZLbGhCmj
788K        file_tree/KqOWv/MH/zWG/8puxfjS
```

Oppgave 1.2

Løsning:

```
find $1 -type f | xargs grep -n --color=always $2 \
|| echo No files containing \"$2\" found.
```

find finner filer (samme som oppagaven over med "-type f"-flagget), og sender treffene separert med linjeskift til stdout. Pipen fanger stdout og peker den til stdin for xargs, som i sin tur plasserer treffet i `grep` sin stdin. `grep`-flaggene `-n` og `-color=always` gjør at grep hendholdsvis viser linjenummer og farger treffene slik som vist i kjøreeksempelet.

De første to strekene i linje to er det man forventer fra andre språk, altså en `OR`. Den siste linja vil altså bli kjørt bare hvis `grep` ikke får noen treff, fordi `grep` da vil ha en returkode større enn 0. Alle "gode" bash-programmer returnerer en statuskode som indikerer om operasjonen gikk bra. I `grep` sitt tilfelle, er "bra" at man får treff, og da vil `grep` returnere 0. Hvis det skjer, vil ikke den siste linja kjøre, fordi vi fikk treff.

Kjøring

```
11:50 $ bash find_word.sh file_tree bil
file_tree/_CVcim:2006:eW68EPmXRbilACbpN
file_tree/fJgme5F:2971:HwbilR0c7PtSZ7fiUdc80q6jf3DIbS9_Kq9fe
file_tree/Kq0Wv/MH/7GvTL2y:4284:FbilumBAFScZqD3ih0_
file_tree/Kq0Wv/MH/_0j2c0QA:7674:suJ19WkYf4_juYYVu4FbilL

11:51 $ bash find_word.sh file_tree java
No files containing "java" found.
```

Oppgave 1.3

Løsning:

```
path=$1
size=$2

files_to_delete=$(find $path -size +${size}k -print)

if [ -n "$files_to_delete" ]
then
  echo Deleting...
  echo "$files_to_delete"
  rm -- $files_to_delete
else
```

```
  echo No files of size $size kilobytes or larger found.
fi

exit 0
```

Først fanger jeg output fra `find` i variabelen `files_to_delete` ved hjelp av *variabelsubstitusjon*. `-size` begrenser søket til en gitt størrelse, plussen betyr *større enn*, og *k* er for kilobyte. Så blir det: *hvis* `-n` (notempty) `$files_to_delete` så: output "Deleting", filer som skal slettes, og kjør kommandoen.
ellers: output "No files [...]".

Med `--` etter en kommando (eller `rm` i eksempelet) sier man at det ikke kommer flere flagg, som i dette tilfellet betyr at programmet ikke tryner på filer som *-testfil.txt*.

Kjøring

```
16:59 $ bash sized_delete.sh file_tree 80
Deleting...
file_tree/_CVcim
file_tree/fJgme5F
file_tree/Kq0Wv/MH/7GvTL2y
file_tree/Kq0Wv/MH/_Oj2c0QA
file_tree/Kq0Wv/MH/gBwNRP
file_tree/Kq0Wv/MH/XhdhBbk
file_tree/Kq0Wv/MH/Z9kP8NB


17:00 $ bash sized_delete.sh file_tree 80
No files of size 80 kilobytes or larger found.
```

Oppgave 1.4

Løsning:

```
cat $1 | sort > $2
```

`cat` leser fra førte argument ($1), spytter linjene til sort, som etter sortering sender output til andre argument ($2) fordi det er en `>` i mellom.

Dette tryner selvfølgelig hvis argumentene ikke er filer eller fildescriptorer.

Kjøring

```
17:09 $ bash sort_file.sh unsorted_fruits sorted_fruits

17:37 $ cat sorted_fruits
apple
```

```
grape
orange
pear
pineapple
```

Kommentar

I noen av filene står det `#!/usr/local/env bash` i stedet for det vanlige `#!/usr/bin`. Dette er fordi jeg jobber på mac som kjører bash3.2 og mangler de kule gnu-verktøya, og derfor trenger jeg å hente ting fra andre steder enn /bin/bash. Sånn som det står nå vil det virke hos andre også.

Oppgave 2

Kjøring

En prøvekjøring kan gjøres ved å kjøre kommandoen

```
cd python
python my_generate_file_tree.py file_tree 10 10 --rec-depth 4
```

Argumentparsing

I stedet for metoden som var foreslått i startfila har jeg brukt argparse til å holde styr på argumenter.

Skriv

```
python my_generate_file_tree -h
```

for å få informasjon om hvordan man setter de forskjellige argumentene.

Funksjoner

Jeg har brukt `os.walk` i stedet for den utdaterte `os.path.walk` Den genererer en tuple du kan bruke i en loop, i stedet for at du sender med callback.

Randomness

Programmet vil stemple filer med tilfeldige atime's og mtime's, generere fra null til <grenseverdi> antall filer og mapper. Bortsett fra det er det random_string som har ansvaret for tilfeldig lengde på fil- og mappenavn samt filinnhold.

Config-objektet

I og med at argumenter som sendes inn via terminalen gjelder hele app'en og det meste av funksjoner avhenger av dette har jeg valgt å legge alle argumentene i et config-objekt.

Jeg mener det gjør koden enklere å ha med å gjøre, og jeg slipper de lange stygge signaturene med mange parametere og default-verdier. I tillegg havner plutselig dokumentasjonen for alt på samme sted, og det er ålreit.

Testing

Jeg har valgt å skrive tester, og disse kan kjøres med for eksempel `py.test -vls`

Kjøring av tester

```
14:44 $ py.test -l
================================= test session starts ==================================
platform darwin -- Python 2.7.5 -- py-1.4.24 -- pytest-2.6.2
collected 12 items

generate_populate_tree_test.py ......
random_string_test.py ......

============================== 12 passed in 1.12 seconds ===============================
```

Kjøring av program

```
11:51 $ python my_generate_file_tree.py file_tree 5 5 --rec-depth 3 -v yes --seed 7653
Creating folder file_tree
Creating folder file_tree/IBYM
Creating folder file_tree/IBYM/_C
Creating folder file_tree/IBYM/_C/FZ
Creating file file_tree/AaRr6
Creating file file_tree/ip40OeguE
Creating file file_tree/eSEzkLN
Creating file file_tree/v
Creating file file_tree/7nsit70RP/l

Created 27 folders and 45 files, for a total of 14893 kilobytes
* Forkortet output
```