1. obligatoriske innlevering, høsten 2014

{Jonathan Feinberg, Joakim Sundnes}
{jonathf,sundnes}@simula.no

September 1, 2014

Innleveringskrav

Det forventes at alle skriptene beskrevet i oppgavene blir skrevet. I tillegg skal en oppsummering skrives i et latex-dokument. Denne skal følge malen gitt på emnesidene (Legges ut 2. september). Innlevering skjer ved opplasting til github. Mer informasjon om klasseromsløsningen vi bruker på github vil bli gitt i god tid før innleveringsfristen.

Resurser som er nyttige for oppgaven blir lagt ut på emnesiden, under linken Obligatoriske oppgaver.

Del 1: Bash

Filtre

Før du begynner, må du pakke ut filtreet:

```
$ tar -xxvf file_tree.tar.gz
```

Obs

Første versjonen av obligen, var dette gjort med et skript. Dette er endret.

Nyttige verktøy

Alle komponenter brukt i oppgavene her må være en del av enten Bash eller standard GNU/Linux-biblioteket. Til hjelp nevner vi følgende tilgjengelige verktøy:

`cat`	Concatenate files and print on the standard output.
`du`	Estimate file space usage.
`echo`	Display a line of text.
`find`	Search file tree for files.
`grep`	Print lines matching a pattern.
`rm`	Remove files or directories.
`sort`	Sort lines of text files.
`wc`	Print newline, word, and byte counts for each file.
`xargs`	Build and execute command lines from standard input.

Husk å gjøre skript eksekvebare før de kjøres:

```
$ chmod +x skript.sh
```

Oppgave 1.1

List opp alle filer som er modifisert de siste `n` dagene. La utskriften inkludere og være sortet på filstørrelse. For eksempel:

```
$ ./list_new_files.sh filetree/ 80
36K     file_tree/Pkvye/htZiVgRE
48K     file_tree/Kq0Wv/5RYWI5kQ
104K    file_tree/Kq0Wv/MH/7GvTL2y
184K    file_tree/zg/grYxji7
564K    file_tree/zg/Hu/dNmOlK
644K    file_tree/Kq0Wv/MH/XhdhBbk
```

Merk at utskriften vil forandre seg med hvilken dag en kjøres.

Oppgave 1.2

Find alle filer som inneholder et gitt ord. Merk at vi snakker om på innsiden av filen, ikke filnavnet. For eksempel, ved bruk av ordene "what" og "hallo" får vi følgende:

```
$ ./find_word.sh file_tree/ what
file_tree/Pkvye/vlfN/ZLbGhCmj:8687:NDa6gmZswhat77iTUFuoNiG23Y
$ ./find_word.sh filetree/ hello
No files containing "hello" found.
```

Oppgave 1.3

Slett alle filer i filtreet med størrelse større enn en gitt verdi. Størrelsen er gitt i kilobyte. Print ut navnene på filene som slettes. For eksempel:

```
$ ./sized_delete.sh file_tree 750
Deleting...
file_tree/zg/Hu/vv/2KKnyIt5
file_tree/Pkvye/vlfN/ZLbGhCmj
file_tree/Kq0Wv/MH/zWG/8puxfjS
file_tree/Kq0Wv/MH/Z9kP8NB
$ ./sized_delete.sh file_tree 750
No files of size 750 kilobytes or larger found.
```

Hint

Back-ticks tillater at man utfører en del-komando:

```
count=‘<bash command>‘
```

Dette kan være nyttig å ha for å for eksempel identifisere om man har 0 eller flere filer som passer størrelseskriteriet.

Oppgave 1.4

Sorter linjene i en fil og lagre dem i en ny fil. For eksempel:

```
$ sh sort_file.sh unsorted_fruits sorted_fruits
$ cat sorted_fruits
apple
grape
orange
pear
pineapple
```

Del 2: Python

I denne oppgaven skal dere skrive et Python skript som skaper et filtre ikke veldig ulikt vedlagt i tar-formatet. Skriptet skal inneholde følgende egenskaper:

- Skriptet skal kunne skrives fra Bash/Cmd som et eget program og skal kreve de følgende tre argumentene: `target`, `folders` and `files`.
- Det skal også være mulig å inkludere de følgende argumentene: `size`, `rec_depth`, `start`, `end`, `seed` and `verbose`
- `target` bestemmer mappen hvor treet skal bygges.
- `rec_depth` bestemmer nivået undermappene får lov til å gå.
- Hvis `verbose` er inkludert (med hvilken som helst verdi), skal skriptet skrive ut de stegene den tar.
- Skriptet skal være tilfeldig (random) i de fleste aspekter: tilfeldig fil- og mappenavn, tilfeldig filinnhold, tilfeldig aksesert og modifisert tidstempling (atime & mtime), og tilfeldig filstørrelse.
- Hvis man inkluderer `seed` som en numerisk verdi, skal skriptet være deterministisk. Mao. skriptet skal gi samme resultat hvis repetert. Dette kan oppnås ved å bruke `random.seed` funksjonen og kun hente tilfeldige elementer fra `random` modulen.
- Argumentene `dirs`, `files`, `size`, `start` og `end` er alle grenseverdier. Hver mappe skal ha $d \in$`[0,dirs]` undermapper (gitt at `rec_depth` ikke er nådd), og $f \in$`[0,files]` filer. Hver fil skal inneholde $s \in$`[1,size*1024]` karakterer
- Markørene for både aksesert tid (atime) og modifisert tid (mtime) skal ha $t \in$`start,end`. Atime og mtime skal som hovedregel være ulike. Datoen skal for enkelthetens skyld være formatert i Unix/Epoch tid notert i sekunder.

Denne oppgaven skal løses platformsuavhengig. Med andre ord, vil vi ikke akseptere løsninger som tilkaller Bash, som `os.system` eller `Popen`.

Som hjelp kan du bruke malen: `my_generate_file_tree.py`. Modifiser den etter behov, men husk å endre dokumentasjon deretter.