DVG001
Introduktion till Linux och små nätverk
Laboration 2

Jonas Sjöberg
860224-xxxx
Högskolan i Gävle
tel12jsg@student.hig.se
https://github.com/jonasjberg

Utförd: 2016-03-03 – 2016-03-06
Kursansvarig lärare: Anders Jackson

Sammanfattning

Laboration i kursen *DVG001 – Introduktion till Linux och små nätverk* som läses på distans via Högskolan i Gävle under vårterminen 2016. Laborationen behandlar skapandet av ett enklare skalprogram eller `bash`-skript. Uppgiften innefattar en rad vanliga kommandon och uppgifter som ingår i administration av `*nix`-system.

Innehåll

Figurer

Tabeller

Programlistningar

Tabell 1: Efterfrågad katalogstruktur

Sökväg	Beskrivning
`inlupp.sh`	ett program
`filett.txt`	innehåll: ”`fil ett`”
`laborationett/`	
`laborationett/filtvaa.txt`	innehåll: ”`fil två`”
`laborationett/filtree.txt`	innehåll: ”`fil tre`”
`laborationett/katalogen/`	
`laborationett/katalogen/skalpgm.sh`	ett program
`laborationett/katalogen/data.txt`	innehåll: godtycklig text, hitta på något
`laborationett/katalogto/data.txt`	innehåll: godtycklig text, något annat
`laborationett/katalogto/filfyra.txt`	innehåll: ”`fil fyra`”

1 Inledning

Uppgiften går ut på att skriva ett exekverbart `shell`-skript som ska skapa en uppsättning filer, kataloger och ytterligare skript.

Det resultat som efterfrågas är en viss katalogstruktur och visas i Tabell 1.

1.1 Bakgrund

Under laborationen används den utvecklingsmiljö som skapades i föregående laboration.

1.2 Syfte

Syftet med uppgiften är att ge tillfälle att öva på skalprogrammering som är en mycket viktig färdighet vid administration av mer avancerade datorsystem. Persondatorer kan många gånger kontrolleras hjälpligt från ett rent grafiskt gränssnitt men större servrar och system kräver ofta kontroll från ett kommandoradsgränssnitt, där skalprogram är avgörande för effektiv hantering av systemet. Det är därför avgörande att ha bra koll på skalprogrammering. Eftersom att `bash` är väldigt vanligt förekommande och dessutom har en del gemensamt med övriga skalprogram så är det lämpligt att läras sig skriva programkod som körs med `bash` ([1]).

2 Planering

Den virtuella maskinen som skapades och installerades i föregående laboration används i befintligt skick, med undantaget att uppdateringar installerats med kommandot i Programlistning 1.

```
1 jonas@debian:~/Documents$
2 sudo apt-get update && sudo apt-get upgrade --assume-yes
```

Programlistning 1: Kommando för att uppdatera paketlistor och installera uppdateringar

Det önskade slutresultatet kan också visualiseras enligt Listning 2. Kommandot `tree` används för att visa innehållet i en katalog som skapats för att efterlikna den slutgiltiga katalogstruktur som programmet ska skapa. Vid utveckling av programmet så jämförs ”modellen” av

den färdiga katalogstrukturen med vad programmet faktiskt skapar, likt moderna metodologier från *TDD – testdriven utveckling*.

```
jonas@debian:~/Documents/lab2/test/target$ tree --charset ascii
.
|-- filett.txt
|-- inlupp.sh
`-- laborationett
    |-- filtree.txt
    |-- filtvaa.txt
    |-- katalogen
    |   |-- data.txt
    |   `-- skalpgm.sh
    `-- katalogto
        |-- data.txt
        `-- filfyra.txt

3 directories, 8 files
```

Programlistning 2: Önskat slutresultat efter körning av `inlupp.sh`.

2.1 Arbetsmetod

Nedan följer en preliminär redogörelse för den experimentuppställning som används under laborationen:

- Laborationen utförs på en `ProBook-6545b` laptop som kör `Xubuntu 15.10` på `Linux 3.19.0-28-generic`.
- Rapporten skrivs i LaTeX som kompileras till pdf med `latexmk`.
- Både rapporten och koden skrivs med texteditorn `Vim`.
- För versionshantering av både rapporten och programkod används `Git`.
- Virtualisering sker med `Oracle VirtualBox` version `5.0.10_Ubuntu r104061`.

2.1.1 Versionshantering

Koden hålls under versionshantering av `Git`. Själva arbetet sker genom att projektets ”repository” klonas in i Debian-maskinen med kommandot i Programlistning 3:

```
jonas@debian:~/Documents$
git clone ssh://spock@192.168.1.107/home/spock/Dropbox/HIG/DVG001_Linux-nätverk/lab/lab2/
```

Programlistning 3: Kommando för att hämta in projektet från värdsystemet

2.1.2 SSH och Vim

Därefter sköts arbetet med koden genom en `SSH`-anslutning in i gästsystemet. All kod skrivs i text-editorn `Vim`.

För att testa koden används ytterligare ett program, `run-tesh.sh` som återfinns i Programlistning 4.2. En körning startas med tangentkombinationen
`m`. För att förknippa tangentkombinationen med en händelse används följande kommando i `Vim`:

```
:map <leader>m :w!\|!./run-test.sh<cr>
```

Det gör att tangentkombinationen `\m` binds till följande handlingar:

1. Spara den aktuella buffern (`:w!`)
2. Exekvera ett externt kommando (`!./run-test.sh`)
3. Simulera en tryckning på enter (`<cr>`)

Även om det bara sparar någon enstaka sekund per körning så är det värt att lära sig hantera verktygen på det här sättet för att spara in mycket tid i långa loppet.

När utvecklingsmiljön är konfigurerad på ett sätt jag trivs med så går det mycket fort att utveckla programmet.

3 Genomförande

3.1 Konfiguration av utvecklingsmiljön

Den första delen av arbetet bestod av att konfigurera utvecklingsmiljön För att kunna installera program och sköta andra administrativa sysslor konfigureras `sudo` enligt följande:

1. Användaren läggs till i `sudo`-gruppen med kommandot: *adduserjonassudo*
2. Konfigurationsfilen för `sudo` ändras. Genom att köra följande kommandon så öppnar en särskild texteditor som tillåter en säkrare mijö för att ändra i filen `/etc/sudoers`:

   ```
   $ su
   $ visudo
   ```

   Med texteditorn lägger man till raden: `jonas ALL=(ALL:ALL) ALL`

   Det ger användaren `jonas` möjlighet att köra alla kommandon som administratör med hjälp av `sudo`.

Sedan byttes nätverkstypen i `VirtualBox` till bridged så att gästsystemet får en egen IP-adress på nätverket. Det underlättar anslutning med `SSH`, som installerades och konfigurerades med kommandot i Programlistning 4.

```
1 jonas@debian:~/Documents$
2 sudo apt-get update && sudo apt-get install vim git
```

Programlistning 4: Kommando för att uppdatera paketlistor och installera texteditorn `vim` och versionshanteringsprogrammet `git`

4 Källkod

4.1 Filen `inlupp.sh`

Filen `inlupp.sh` är det primära resultatet av uppgiften. Filens innehåll återfinns i Programlistning 4.1.

```
#!/usr/bin/env bash
# ------------------------------------------------------------------------------
#
# DVG001 -- Introduktion till Linux och små nätverk
#                              Inlämningsuppgift #2
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Author:   Jonas Sjöberg
#           tel12jsg@student.hig.se
#
# Date:     2016-03-03 -- 2016-03-06
#
# License:  Creative Commons Attribution 4.0 International (CC BY 4.0)
#           <http://creativecommons.org/licenses/by/4.0/legalcode>
#           See LICENSE.md for additional licensing information.
# ------------------------------------------------------------------------------

set -e                      # Avbryt om ett kommando returnerar fel (nollskiljt)
#set -x                     # Avkommentera för debug-läge

SCRIPT_NAME=$(basename $0) # Den här filens namn, utan fullständig sökväg.
SCRIPT_DIR=$(dirname $0)   # Katalogen som den här filen ligger i.


# Funktion 'msg_error'
# Skriver ut meddelanden till stdout och stderror.
function msg_error()
{
    printf "${SCRIPT_NAME} [ERROR] : %s\n" "$*" 2>&1
}

# Funktion 'create_file'
# Skapar en fil med ett visst innehåll.
# Tar två argument: sökväg till målfilen och innehållet som ska användas.
function create_file()
{
    # Avbryt om antalet argument inte är två.
    if [ $# -ne 2 ]
    then
        return 1
    fi

    # Variabelutbyte sätter variabeln dest till det som står mellan
    # minustecknet och högerklammern om variabeln 1 är tom.
    #
    #   ${parameter:-word}
    #   If parameter is unset or null, the expansion of word is substituted.
    #   Otherwise, the value of parameter is substituted.
    #
    # https://www.gnu.org/software/bash/manual/html_node/Shell-Parameter-Expansion.html
    local dest="${1:-}"
    local cont="${2:-}"

    # Avbryt om variabeln dest är tom/null.
    if [ -z "$dest" ]
    then
        msg_error "Missing argument"
        return 1
    fi

```

```
60      # Avbryt om målfilen redan existerar.
61      if [ -e "$dest" ]
62      then
63          msg_error "Destination already exists"
64          return 1
65      else
66
67          # Testa om det finns skrivrättigheter för målfilen.
68          if [ -w "$dest" ]
69          then
70              msg_error "Need write permissions for destination"
71              return 1
72          else
73              # Skriv innehållet i variabeln "cont" till sökvägen/filnamnet i
74              # variabeln "dest".
75              echo "$cont" > "$dest"
76
77              # Använd returvärdet från echo-operationen som returvärde för
78              # funktionen 'create_fil_ett'. Kan användas för felkontroll, där
79              # nollskiljt returvärde är någon typ av fel.
80              return $?
81          fi
82      fi
83  }
84
85  # Funktion 'create_dir'
86  # Skapar en katalog. Tar ett argument.
87  function create_dir()
88  {
89      local dest="${1:-}"
90
91      # Avbryt om argumentet dest är tomt/null.
92      if [ -z "$dest" ]
93      then
94          return 1
95      else
96          # Flaggan '-p' skapar om nödvändigt underkataloger till målkatalogen.
97          mkdir -p "$dest"
98      fi
99  }
100
101
102 # Lagra de kataloger som ska skapas i variabler d1, d2 och d3.
103 d1="${SCRIPT_DIR}/laborationett"
104 d2="${d1}/katalogen"
105 d3="${d1}/katalogto"
106
107
108 # Skapa filen 'filtvaa.txt' med funktionen "create_file".
109 create_file 'fil två' "${d1}/filtvaa.txt"
110
111
112 # Skapa katalog och två filer.
113 create_dir "$d1"
114 create_file "${d1}/filtvaa.txt" 'fil två'
115 create_file "${d1}/filtree.txt" 'fil tre'
116
117
118 # Skapa katalog och filen 'skalpgm.sh'
119 # EOF är omringat med '' för att allt i "here-dokumentet" ska tolkas ordgrant
```

```
120 # och inte ersättas med t.ex. variablers värden eller output från "command
121 # substitution".
122 create_dir "${d2}"
123 cat << 'EOF' > "${d2}/skalpgm.sh"
124 #!/usr/bin/env bash
125 # Räknar antal rader i './data.txt' och '../katalogto/data.txt'
126
127 # Hämta full sökväg till scriptet under många omständigheter.
128 # Källa: http://stackoverflow.com/a/246128
129 SCRIPT_DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"
130
131 # Se till att vara i rätt katalog för att nedanstående cat-kommando
132 # funkar med relativa sökvägar till de båda 'data.txt'-filerna.
133 cd "$SCRIPT_DIR" && cd ..
134
135 # Använd "command substitution" för att spara resultatet av pipelinen
136 # till variabeln "antal_rader".
137 antal_rader=$(cat "katalogen/data.txt" "katalogto/data.txt" | wc -l)
138
139 # Skriv ut resultatet med printf.
140 # Referens: "Pro Bash Programming: Scripting the GNU/Linux Shell"
141 #           Johnson, Chris and Varma, Jayant. 2nd Edition. Apress 2015.
142 printf "\n%s %s\n" "totalt antal rader:" "$antal_rader"
143 cd --
144 EOF
145
146
147 # Gör programmet 'skalpgm.sh' exekverbart.
148 chmod +x "${d2}/skalpgm.sh"
149
150
151 # Spara godtycklig text i filen "data.txt" med ett "here document".
152 # Referens: Advanced Bash-Scripting Guide
153 #           http://tldp.org/LDP/abs/html/here-docs.html
154 cat << EOF > "${d2}/data.txt"
155 BEOWULF.
156
157
158
159
160 I.
161
162 THE LIFE AND DEATH OF SCYLD.
163
164
165 {The famous race of Spear-Danes.}
166
167           Lo! the Spear-Danes' glory through splendid achievements
168           The folk-kings' former fame we have heard of,
169           How princes displayed then their prowess-in-battle.
170
171 {Scyld, their mighty king, in honor of whom they are often called
172 Scyldings. He is the great-grandfather of Hrothgar, so prominent in the
173 poem.}
174
175           Oft Scyld the Scefing from scathers in numbers
176         5 From many a people their mead-benches tore.
177           Since first he found him friendless and wretched,
178           The earl had had terror: comfort he got for it,
179           Waxed 'neath the welkin, world-honor gained,
```

```
180           Till all his neighbors o'er sea were compelled to
181         10 Bow to his bidding and bring him their tribute:
182           An excellent atheling! After was borne him
183
184 {A son is born to him, who receives the name of Beowulf--a name afterwards
185 made so famous by the hero of the poem.}
186 EOF
187
188
189 # Skapa katalog och fil "data.txt" med godtyckligt innehåll.
190 create_dir "${d3}"
191 cat << EOF > "${d3}/data.txt"
192 II.
193
194 SCYLD'S SUCCESSORS.--HROTHGAR'S GREAT MEAD-HALL.
195
196
197 {Beowulf succeeds his father Scyld}
198
199           In the boroughs then Beowulf, bairn of the Scyldings,
200           Belovèd land-prince, for long-lasting season
201           Was famed mid the folk (his father departed,
202           The prince from his dwelling), till afterward sprang
203         5 Great-minded Healfdene; the Danes in his lifetime
204           He graciously governed, grim-mooded, agèd
205 EOF
206
207
208 # Skapa filfyra.txt med innehåll "fil fyra"
209 #echo 'fil fyra' > "${d3}/filfyra.txt"
210 create_file "${d3}/filfyra.txt" 'fil fyra'
211
212
213 # Avsluta programmet.
214 # Returnerar koden från det senast avslutade kommandot.
215 exit $?
```

Källkod för filen `inlupp.sh`

4.2 Filen `run-test.sh`

Filen `run-test.sh` användes för att testa programmet under utveckling. Filens innehåll återfinns i Programlistning 4.2. Exempel på körning av `inlupp.sh` med testprogrammet `run-test.sh` finns i Figur 1 och 2.

```
1  #!/usr/bin/env bash
2  # ------------------------------------------------------------------------
3  #
4  # DVG001 -- Introduktion till Linux och små nätverk
5  #                              Inlämningsuppgift #2
6  # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
7  # Author:   Jonas Sjöberg
8  #           tel12jsg@student.hig.se
9  #
10 # Date:     2016-03-03 -- 2016-03-06
11 #
12 # License:  Creative Commons Attribution 4.0 International (CC BY 4.0)
13 #           <http://creativecommons.org/licenses/by/4.0/legalcode>
```

```
#           See LICENSE.md for additional licensing information.
# ------------------------------------------------------------------------------

#set -e                     # Avbryt om ett kommando returnerar fel (nollskiljt)
#set -x                     # Avkommentera för debug-läge

TEST_DIR="run_test"
SRC_DIR="../src"

C_NORMAL=$(tput sgr0)
C_RED=$(tput setaf 1)
C_GREEN=$(tput setaf 2)
C_YELLOW=$(tput setaf 3)
C_BLUE=$(tput setaf 4)
HALFTAB='  '
FULLTAB='    '


# FUNCTION MSG_TYPE() -- Prints timestamped debug messages
# Messages can be of three different types (severity): ERROR, INFO and WARNING.
# Usage: msg_type info example informative text
#        msg_type warn some unexpected thing happened
#        msg_type "just text, no type specified"
function msg_type()
{
    if [ $# -gt 1 ]
    then
        local type="$1"
        shift
    fi

    local text="$*"

    case "$type" in
        error ) color=$C_RED
                label="ERROR"   ;;
        info  ) color=$C_GREEN
                label="INFO"    ;;
        warn  ) color=$C_YELLOW
                label="WARNING" ;;
        *     ) color=$C_NORMAL
                label=""        ;;
    esac

    # Surround label with brackets if not empty.
    if [ ! -z "$label" ]
    then
        label="${color}[${label}]${C_NORMAL}"
    fi

    TIMESTAMP="$(date +%H:%M:%S)"

    printf "${C_BLUE}[${TIMESTAMP}]${C_NORMAL} ${label}"
    printf "${FULLTAB}%s\n" "$text"

    #time box text
}


```

```
74 msg_type info "Starting test runner"
75 msg_type info "Setting up test environment"
76
77 if [ -x "$TEST_DIR" ]
78 then
79     msg_type warn "Removing previous test environment"
80     rm -rfv "$TEST_DIR"
81 fi
82
83 mkdir -v "$TEST_DIR"
84 cp -v "${SRC_DIR}/inlupp.sh" "$TEST_DIR"
85
86
87 msg_type info "STARTING TEST NOW"
88
89 # Anvand ett sub-shell for att isolera "cd"-kommandot.
90 (
91     cd $TEST_DIR && ./inlupp.sh || msg_type error "Failed to start"
92
93     msg_type info "Done running script .."
94     msg_type info "Contents of "$(pwd)":"
95     tree
96
97     for f in $(find . -type f -not -name "inlupp.sh")
98     do
99         msg_type info "File: ${f}"
100        msg_type info "Contents:"
101        cat "$f"
102        echo ""
103    done
104
105    msg_type info "Running \"skalpgm.sh\""
106    cd "laborationett/katalogen" || msg_type error "Failed changing directory"
107    ./skalpgm.sh || msg_type error "Failed executing \"skalpgm.sh\""
108 )
109
110
111 msg_type info "Done running test"
```

Källkod för `run-test.sh`, program som kör `inlupp.sh` och skriver ut resultat av körningen.

```
[02:42:22] [INFO]    Starting test runner
[02:42:22] [INFO]    Setting up test environment
[02:42:22] [WARNING]    Removing previous test environment
removed ‘run_test/laborationett/katalogto/filfyra.txt’
removed ‘run_test/laborationett/katalogto/data.txt’
removed directory: ‘run_test/laborationett/katalogto’
removed ‘run_test/laborationett/katalogen/skalpgm.sh’
removed ‘run_test/laborationett/katalogen/data.txt’
removed directory: ‘run_test/laborationett/katalogen’
removed ‘run_test/laborationett/filtvaa.txt’
removed ‘run_test/laborationett/filtree.txt’
removed directory: ‘run_test/laborationett’
removed ‘run_test/inlupp.sh’
removed ‘run_test/filett.txt’
removed directory: ‘run_test’
mkdir: created directory ‘run_test’
‘../src/inlupp.sh’ -> ‘run_test/inlupp.sh’
[02:42:22] [INFO]    STARTING TEST NOW
mkdir: created directory ‘./laborationett’
mkdir: created directory ‘./laborationett/katalogen’
[02:42:22] [INFO]    Done running script ..
[02:42:22] [INFO]    Contents of /home/jonas/Documents/lab2/test/run_test:
.
├── filett.txt
├── inlupp.sh
└── laborationett
    ├── filtree.txt
    ├── filtvaa.txt
    ├── katalogen
    │   ├── data.txt
    │   └── skalpgm.sh
    └── katalogto
        ├── data.txt
        └── filfyra.txt

3 directories, 8 files
[02:42:22] [INFO]    File: ./laborationett/katalogto/filfyra.txt
[02:42:22] [INFO]    Contents:
fil fyra

[02:42:22] [INFO]    File: ./laborationett/katalogto/data.txt
[02:42:22] [INFO]    Contents:
II.

SCYLD'S SUCCESSORS.--HROTHGAR'S GREAT MEAD-HALL.


{Beowulf succeeds his father Scyld}

          In the boroughs then Beowulf, bairn of the Scyldings,
          Belovèd land-prince, for long-lasting season
          Was famed mid the folk (his father departed,
          The prince from his dwelling), till afterward sprang
        5 Great-minded Healfdene; the Danes in his lifetime
          He graciously governed, grim-mooded, agèd

[02:42:22] [INFO]    File: ./laborationett/katalogen/skalpgm.sh
[02:42:22] [INFO]    Contents:
#!/usr/bin/env bash
# Räknar antal rader i './data.txt' och '../katalogto/data.txt'

# Hämta full sökväg till scriptet under många omständigheter.
# Källa: http://stackoverflow.com/a/246128
SCRIPT_DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"

# Se till att vara i rätt katalog för att nedanstående cat-kommando
# funkar med relativa sökvägar till de båda 'data.txt'-filerna.
cd "$SCRIPT_DIR" && cd ..

# Använd "command substitution" för att spara resultatet av pipelinen
# till variabeln "antal_rader".
antal_rader=$(cat "katalogen/data.txt" "katalogto/data.txt" | wc -l)

# Skriv ut resultatet med printf.
# Referens: "Pro Bash Programming: Scripting the GNU/Linux Shell"
#           Johnson, Chris and Varma, Jayant. 2nd Edition. Apress 2015.
printf "\n%s %s\n" "totalt antal rader:" "$antal_rader"
cd --
```

Figur 1: Exempel på körning av testprogrammet `run-test.sh`. (1/2)

```
[02:42:22] [INFO]    File: ./laborationett/katalogen/data.txt
[02:42:22] [INFO]    Contents:
BEOWULF.



I.

THE LIFE AND DEATH OF SCYLD.


{The famous race of Spear-Danes.}

          Lo! the Spear-Danes' glory through splendid achievements
          The folk-kings' former fame we have heard of,
          How princes displayed then their prowess-in-battle.

{Scyld, their mighty king, in honor of whom they are often called
Scyldings. He is the great-grandfather of Hrothgar, so prominent in the
poem.}

          Oft Scyld the Scefing from scathers in numbers
        5 From many a people their mead-benches tore.
          Since first he found him friendless and wretched,
          The earl had had terror: comfort he got for it,
          Waxed 'neath the welkin, world-honor gained,
          Till all his neighbors o'er sea were compelled to
       10 Bow to his bidding and bring him their tribute:
          An excellent atheling! After was borne him

{A son is born to him, who receives the name of Beowulf--a name afterwards
made so famous by the hero of the poem.}

[02:42:22] [INFO]    File: ./laborationett/filtvaa.txt
[02:42:22] [INFO]    Contents:
fil två

[02:42:22] [INFO]    File: ./laborationett/filtree.txt
[02:42:22] [INFO]    Contents:
fil tre

[02:42:22] [INFO]    File: ./filett.txt
[02:42:22] [INFO]    Contents:
fil ett

[02:42:22] [INFO]    Running "skalpgm.sh"

totalt antal rader: 44
[02:42:22] [INFO]    Done running test
```

Figur 2: Exempel på körning av testprogrammet `run-test.sh`. (2/2)

5 Resultat

Resultatet är ett fungerande program som möter de krav som ställts. Programmet är delvis utökat med extra funktionalitet, däribland test för vanliga fel och diverse specialfall (”edge-cases”) men programmet kan fortfarande förbättras på många sätt. Samtidigt så kan den här typen av skalprogram vara lämpliga till att sköta enklare uppgifter, särskilt enkla operationer som skapande av kataloger och filer bör kanske inte stoppas i funktioner och kompliceras i onödan.

Jag vill hävda att mitt program möter den funktionalitet som efterfrågats. Programmet är också skrivet på ett begripligt sätt och innehåller en del säkerhetsfunktioner. Dock skulle programmet kunna härdas och förbättras på många sätt, men i den här typen av kurs är jag osäker på vart det är dags att dra gränsen och vad som förväntas.

Till exempel använder jag funktioner, som inte tagits upp i föreläsningarna. Dessutom använder jag oftare `printf` än `echo`, något som jag antar också räknas som överkurs i det här sammanhanget. Baserat på *echo, and why you should avoid it* i [2])

6 Diskussion

Bra exempel på stabil och välskrivna `bash`-skript är de som körs som en del av boot-processen i Debian och Ubuntu. Delar av mitt enkla testprogram `run-test.sh` har inspirerats av kod som körs under bootsekvensen av Debian och ligger bakom de färgade statusmeddelanden som skrivs ut. I Debian `8.3` (jessie) finns programmet i sökvägen
`/lib/lsb/init-functions.d/20-left-info-blocks`.

Friheten att kunna ta reda på hur en funktion är implementerad i ett stort och avancerat system som Debian, av utvecklare som (förhoppningsvis) vet vad de gör är en del av vad som gör öppen och fri programvara överlägsen för studenter av mjukvaruutveckling.

En frågeställning som verkar återkommande är den om ”best practice” inom portabla skalprogram. Då jag söker efter råd och svar finns många gånger referenser till `POSIX`-standarden [3]–[6] och samtidigt verkar många i praktiken göra kompromisser med användning av standardiserade och portabla metoder/program beroende på användningsområde.

Eftersom att `bash` är så pass vanligt så verkar många så kallade ”bashisms” [7] användas bland ”best practice” lösningar. Jag antar att det svaret kan ligga någonstans mellan praktikalitet och perfektionism, beroende på plattform, användningsområdet, etc.

7 Slutsatser

Jag skriver ofta små program i `bash` eller `python` för att lösa olika problem som dyker upp i min vardagliga datoranvändning, eller helt enkelt för att automatisera repetitiva uppgifter.

Det kan vara svårt att förklara för ”oinvigda” hur pass kraftfullt ett sådant arbetssätt är. Den frihet som det innebär att själv kunna utveckla lösningar på problem, på ett relativt enkelt sätt och på kort tid, är otroligt värdefull.

`Unix`-liknande system i allmänhet ser ökad popularitet i och med så kallade ”internet of things”, mobiltelefoner och andra datorplattformar konstruerade kring någon typ av `unix`-inspirerad bas. Även innan det så har de alltid utgjort stommen av de största systemen och nätverken. Arbetsmarknaden verkar behöva duktiga systemadministratörer särskilt erfarna av `*nix`-system och jag misstänker att en riktigt duktig och erfaren `bash`-programmerare alltid kan hitta jobb.

Referenser

[1] P. Jackson, DVG001 — *introduktion till linux och små nätverk, föreläsning 2: Skalprogrammering*, Lecture slides, [Online; accessed 13-February-2016], 2015. URL: `https://lms.hig.se/bbcswebdav/pid-382566-dt-content-rid-1646684_1/courses/DVG001.28401.2016/F2.pdf`.

[2] C. Johnson och J. Varma, *Pro Bash Programming, Second Edition: Scripting the GNU/-Linux Shell*, 2nd. Berkely, CA, USA: Apress, 2015, ISBN: 1484201221, 9781484201220.

[3] IEEE, *IEEE Std 1003.1-2001 Standard for Information Technology — Portable Operating System Interface (POSIX) Base Definitions, Issue 6*. 2001, s. xliv + 448, Revision of IEEE Std 1003.1-1996 and IEEE Std 1003.2-1992) Open Group Technical Standard Base Specifications, Issue 6., ISBN: 1-85912-247-7 (UK), 1-931624-07-0 (US), 0-7381-3047-8 (print), 0-7381-3010-9 (PDF), 0-7381-3129-6 (CD-ROM).

[4] ——, *IEEE Std 1003.1-2001 Standard for Information Technology — Portable Operating System Interface (POSIX) System Interfaces, Issue 6*. 2001, s. xxx + 1690, Revision of IEEE Std 1003.1-1996 and IEEE Std 1003.2-1992) Open Group Technical Standard Base Specifications, Issue 6., ISBN: 1-85912-247-7 (UK), 1-931624-07-0 (US), 0-7381-3094-4 (print), 0-7381-3010-9 (PDF), 0-7381-3129-6 (CD-ROM).

[5] ——, *IEEE Std 1003.1-2001 Standard for Information Technology — Portable Operating System Interface (POSIX) Shell and Utilities, Issue 6*. 2001, s. xxxii + 1090, Revision of IEEE Std 1003.1-1996 and IEEE Std 1003.2-1992) Open Group Technical Standard Base Specifications, Issue 6., ISBN: 1-85912-247-7 (UK), 1-931624-07-0 (US), 0-7381-3050-8 (print), 0-7381-3010-9 (PDF), 0-7381-3129-6 (CD-ROM).

[6] ——, *IEEE Std 1003.1-2001 Standard for Information Technology — Portable Operating System Interface (POSIX) Rationale (Informative)*. 2001, s. xxxiv + 310, Revision of IEEE Std 1003.1-1996 and IEEE Std 1003.2-1992) Open Group Technical Standard Base Specifications, Issue 6., ISBN: 1-85912-247-7 (UK), 1-931624-07-0 (US), 0-7381-3048-6 (print), 0-7381-3010-9 (PDF), 0-7381-3129-6 (CD-ROM).

[7] I. team, *Coding style guidelines: Shell script*, [Online; accessed 06-March-2016], 2004-2013. URL: `http://www.inquisitor.ru/doc/coding-style-shell.html`.