Програмиране в UNIX среда

Използване на командeн шел и създаване на скриптове: tcsh, bash, awk, python

Shell programming

- As well as using the shell to run commands you can use its built-in programming language to write your own commands or programs.
- Creating and executing the shell script:
  - Use a text editor to create a file:
    - **emacs filename**
  - Define execute permission:
    - **chmod u=rwx filename**
  - Execute the script
    - **filename**

Shell programming - example

Ø $ cat > lh

```
#list home directory
cd
pwd
ls
^D
```

Ø $ chmod u+x lh

```
$ lh
/usr/home/icc
courses
file.txt
dir1
dir2
$
```

What is with current directory? Will the home directory become the current directory when the script lh finishes?

Different shells for programming

- Ø Possibility of using different shells:
  - Ø Bourne shell - common for all Unix systems - most often used
- Ø First line in the script defines the shell:
  - Ø #!/bin/sh Bourne shell
  - Ø #!/bin/csh C-shell
  - Ø #!/bin/tcsh TC-shell
  - Ø #!/bin/bash BASH shell

Shell programming - Example 1

```
Ø       #!/bin/bash
    # This script displays the date, time,
    # username and current directory.
Ø       echo "Date and time is:"
    date
    echo
    echo "Your username is: $(whoami) \n"
    echo "Your current directory is: \c"
    pwd
```

Ø Output:

```
Ø       Date and time is:
    Mon Feb 27 17:21
Ø       Your username is: icc
    Your current directory is: /home/icc/course/doc
```

Shell programming - Example 2

- Read commands from the terminal and process them in a sub-directory:

```
#!/bin/sh
# usage: process sub-directory
        dir=$(pwd)
for i in *
do
        if  [ -d $dir/$i  ]
           then
           cd $dir/$i
           while echo “$i:”
                      read x
           do
               eval $x
           done
    fi
done
```

The user types the command:
process dir
The user is prompted to supply the name of the command to be read in.
This command is executed using the built-in eval function

Shell programming - Passing arguments

- Passing command arguments to the script: *comm par1 par2*
  - $0 - command name
  - $1 - $9 - parameters
  - Each parameter corresponds to the position of the argument on the command line.
  - $* - all parameters

Passing parameters - examples

- $showpar The first five command line

- echo “First and third parameters are: $1 $3”
- --
  First and third parameters are: The five

- *script printps:*
- #!/bin/sh
  # printps - Convert ASCII files to PostScript # and send them to the PostScript printer
  # Use a local utility "a2ps"

  a2ps $* | lpr -Pps1

- *Executing printps script:*
- $ printps elm.txt vi.ref msg

Shell programming - handling variables

- Special shell variables
  - **Name** **Description**
  - $1 - $9 these variables are the positional parameters.
  - $0 the name of the command
  - $# the number of positional arguments
  - $? the exit status of the last command executed
  - $$ the process number of this shell
  - $! the process id of the command run in the background.
  - $* a string containing all the arguments
  - $@ the same as $* , except when quoted.

Managing more than 9 parameters

Ø shift - shifts arguments: $n+1 becomes $n

Ø Example:

  Ø shift_demo script:

  Ø **echo "arg1=$1 arg2=$2 arg3=$3"**

  Ø **shift**

  Ø **echo "arg1=$1 arg2=$2 arg3=$3"**

  Ø **$ shift_demo one two three four**

  Ø **arg1=one arg2=two arg3=three**
  arg1=two arg2=three arg3=four

Shell programming - handling variables (cont.)

- Definition Description
- $var expand value of the variable var
- ${var} the same as above except the braces enclose the name of the variable to be substituted.
- ${var-val} value of var if var is defined; otherwise val. $var is not set to val.
- ${var=val} value of var if var is defined; otherwise val. If undefined $var is set to val.
- ${var?mess} if defined, $var; otherwise print message and exit the shell. If the message is empty, print a standard message
- ${var+val} val if $var is defined, otherwise nothing.

Note: All variables are of text type

Shell programming – program statements

- Reading user input
  - To read standard input into a shell script use the read command.

```
echo "Please enter your name:"
read name
echo "Welcome to MDH $name"
```

- Conditional statements

```
if [ condition ]
then
        commands            # if condition is true
elif [ condition ]
        commands            # else if
else
        commands            # else
fi
```

IF statement - example

Ø

```
# test if user in logged in
# input: getuser username

user=$1      # input parameter
if who | grep -s $user > /dev/null
then
        echo $user is logged in
else
        echo $user not available
fi
```

Ø

```
#Testing for files and variables:
if [ ! -f $FILE ]; then
        if [ "$WARN" = "yes" ]; then
                echo "$FILE does not exist"
        fi
fi
```

Test conditions:

-e file	true if the file exists
-d file	true if file is a directory
-f file	true if the file is an ordinary file
-L file	true if file is a symbolic link
-r[wx] file	true if the file is readable (writable, executable)
-z str	true if the length of the str is zero
-n str	true if str is not a null str
str1 = str2	true if str1 and str2 are identical
str1 != str2	true if str1 and str2 are not identical
n1 -eq n2	true if numbers n1 and n2 are equal

- Other keywords: -ge, -gt -le, -lt, -ne

Shell programming – program statements

```
Ø  The case statement
Ø       case word in
Ø               pattern1)       command(s)
                        ;;
Ø               pattern2)       command(s)
                        ;;
Ø               patternN)       command(s)
                        ;;
Ø               esac
Ø  The for statement
Ø       for var in list-of-words
   do
        commands
   done
```

Example

```
Ø       #!/bin/bash
    # compare files in two directories
    # input cmpfile dir1 dir2
Ø       dir1=$1
    dir2=$2
Ø       for i in $(ls $dir1)
    do
            echo $i:
            cmp $dir1/$i $dir2/$i
            echo
    done
```

while and until statements

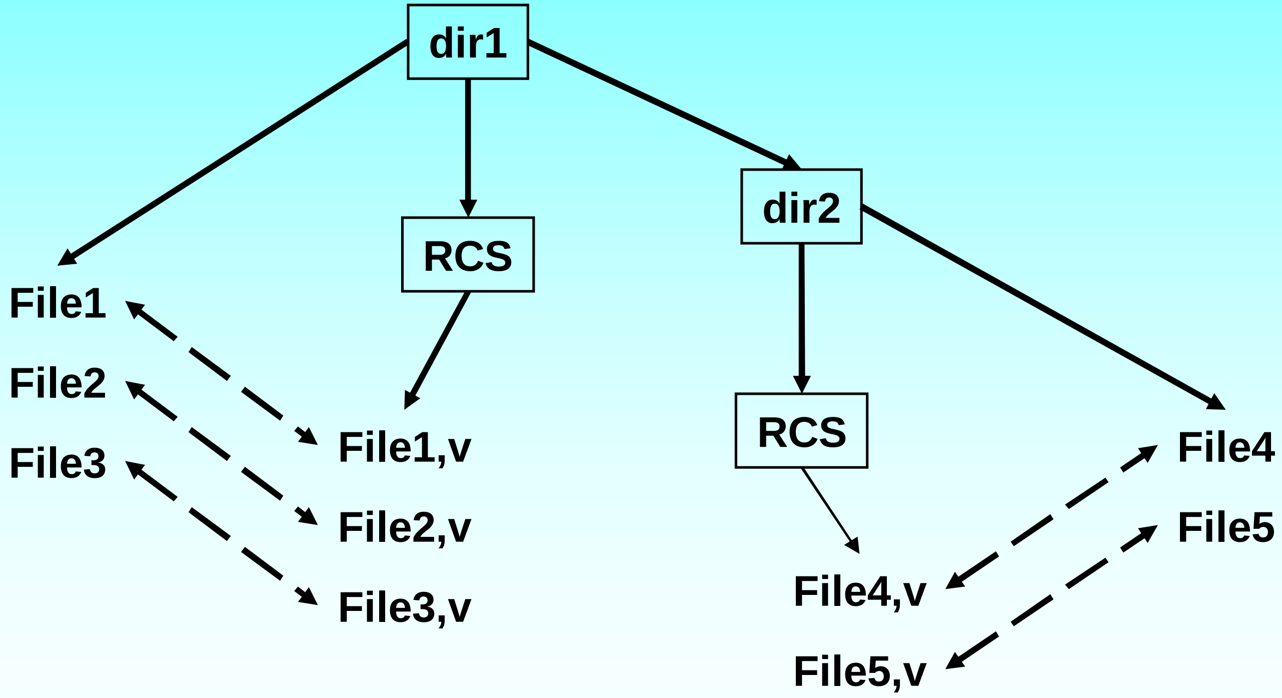
```
Ø        while command-list1
    do
             command-list2
    done

Ø        until command-list1
    do
             command-list2
    done
```

Other statements

- Including text in a shell script
-         # this script outputs the given text before it
  # runs
-         cat << EOF
  This shell script is currently under development, please report any
  problems to me (@uni-sofia.bg)
  EOF
- exec command - executing without creating a new process
-         exec /usr/local/test/bin/test_version

Shell programming - A more complex example

- lsver program
  - list files in a directory tree and compare date of files with the dates of the corresponding files in the underlying RCS directories



lsver man page

- NAME
  lsver - list files and show which are versioned
- SYNOPSIS
  lsver [-r] [name ...]
- DESCRIPTION
  The command lsver shows files in the same way as ls command and in addition it gives information about if files are writable or readonly, and if there exist corresponding RCS files in the RCS directory. If for a file a RCS file is found, then the text Ver denotes it. Date of the file with the date of the RCS file is compared. If the exported file is older then the RCS file, then the text old is displayed, otherwise the texts up-to-date or new are shown.
- PARAMETERS
- name
  denotes a directory or a file. If no name is specified then the current directory is listed.
- OPTIONS
  -r
  lists all subdirectories (corresponding to the ls -R option).

man lsver (cont.)

- EXAMPLES
- lsver

```
RCS                    dir
get_param              -        Ver      up-to-date
ipa_structure          -        Ver       up-to-date
lsver                  w        Ver      new
project                w        Ver      up-to-date
project.help           -        Ver      old
```

- SEE ALSO
- **ls(1), rcs(1)**

Shell programming - lsver program

```
Ø         #!/bin/bash
    #--------------------------------------------------------------
    # lsver - list versioned files
    # $Id 1.2  1993/03/23  13:43:05  litov
    # List files and shows which are versioned
    # command: lsver [-r] file ....
    #--------------------------------------------------------------
Ø         function list {                           # define function list
      ls_arg=$*                                     # take all input parameters
      for arg in $ls_arg; do              # process all parameters
      if [ -d $arg ]; then                # if directory
        dir=$arg                                    # put its name in dir var
      else
        dir=$(dirname $arg)               # take dir part into dir var.
```

Shell programming - lsver program - cont.

```
Ø                      #process files for each parameter
Ø                      for fl in $(ls $arg); do
           file=$(basename $fl)                        # for each file get name
           if [ -d $dir/$file ]; then          # if it is a directory
            rd=dir                                     # specify it
           elif [ -w $dir/$file ]; then                # if it is writable file
            rd=”w”                                     # specify w
           else                                                  # no writable file
        rd=”-”                                                   # specify read-only
           fi
Ø                       # we shall now compare file with the possible RCS/file,v
Ø                       ver=                                               # initialize ver
           age=                                                  # initialize time comp.
           rcs_file=$dir/RCS/${file},v                 # specify RCS file
           if [ -f $rcs_file ]; then                   # if there is RCS file
            ver=“Ver”                                            # specify “Ver”
            age=up-to-date                                       # assume - up-to date
            if [ $dir/$file -ot $rcs_file ]; then  # file older
             age=old
            elif [ $dir/$file -nt $rcs_file ; then  # file newer
             age=new
            fi
           fi
           echo “$file $rd $ver $age”                  # print info about item
          done
     done                                                        # process all params
Ø    } #function list
```

Shell programming - lsver program - cont.

```
Ø         #------------------------------------------------------------
     # main program
     #------------------------------------------------------------
Ø         usage=”Usage: lsver [-r] [names ...]”
Ø         r_flag=false                                       # default option no -r
Ø         for arg in $* ; do                         # process input arguments
          case $arg in
           -r*) r_flag=true              # recursive flag
                    shift ;;                         # skip to next argument
Ø          -*)  echo -u2 $usage                     # illegal option
               exit 1 ;;                 # exit
Ø                   esac
     done


Ø         if [ $r_flag = false ]; then               # if no recursive search
      list “$*” | awk ‘{printf(“%-25s %-3s %-4s %s\n”,$1,$2,$3,$4) }’
```

Shell programming - lsver program - cont.

```
ø       else                                        # recursive
   search
     names=”$@”                                          #
   save all parameters
     if [ -z “$names” ]; then                       # if no
   parameter def.
       names=.                                      # take
   default directory
     fi


Ø    # find tree and pipe to read loop
Ø       find $names -print| while read x; do
```

Литература:

- http://www.wylug.org.uk/talks/2003/04/unix.pdf
- http://ce.sharif.edu/courses/ssc/unix/resources/root/Slides/unixhistory.pdf
- http://www.cs.uga.edu/~eileen/1730/Notes/intro-UNIX.ppt
- http://remus.rutgers.edu/cs416/F01
- http://www.cs.virginia.edu/~cs458/
- http://www.bobbooth.staff.shef.ac.uk/hpcs/materials/material.html
- http://www.comm.utoronto.ca/~jorg/teaching/ece461
- http://home.iitk.ac.in/~navi/sidbilinuxcourse/
- http://www.cs.washington.edu/homes/bershad/Mac/ssh/practicalmagic.pdf
- http://www.cs.cf.ac.uk/Dave/C/CE.html
- http://www.le.ac.uk/cc/tutorials/c/ccccintr.html
- http://www.shef.ac.uk/uni/academic/N-Q/phys/teaching/phy225/index.html