

## Shell Programming

### Examples of simple shell scripts:

[man2pdf](#): converts a man page to pdf

```
#!/bin/sh

ls -lt /usr/share/man/man1/$1.1

troff -man /usr/share/man/man1/$1.1 > t2
echo ..done troff

cat t2 | dpost > t3
echo ..done dpost

ps2pdf t3 $1.pdf
echo ..done ps2pdf
```

[man2pdfs](#): simple version of [man2pdf](#)

```
#!/bin/sh

ls -lt /usr/share/man/man1/$1.1

troff -man /usr/share/man/man1/$1.1 |
    dpost | ps2pdf - $1.pdf
```

**NOTE:** man2pdfs does not use temporary files  
(thus you do not have to worry about permissions to create such files and deleting them at the end)

---

### Introduction to Bourne Shell

for more details see man page for sh (% man sh) (sh.pdf).

## Loop Statements:

```
while list until list for name [ in word ... ]  
do list do list do list  
done done done
```

## Conditional Statements:

```
if list case word in  
then list pattern) list ;;  
list pattern) list ;;  
[ elif list .....  
then list ] ... esac  
[ else list ]  
fi
```

## Misc. Statements:

**test:** Evaluates conditional expressions

**example:** `test $i -le 2`  
returns 0 (true) if \$i is less than or equal to 2.

**expr:** Evaluates arithmetic expression

**example:** `i=`expr $i + 1``  
increments `i` by 1 (like `i++` in C).

**read:** Reads one line from standard input

**example:** `read var < file`  
reads a line from `file` and assign it to `var`

**echo:** Writes to the standard output

**example:** `echo "the line read is:" $var`  
writes: the line read is: <the content of `var`>

**set:** Assigns values to positional parameters `$1`, `$2`, ...

**example:** `set hello world`  
`echo $2`  
  
`world` (the value is of `$2` is: world)

**IFS:** Internal Field Separators

**example:** `IFS=@`  
`set hello@world`  
`echo $2`

**world** (the value is of \$2 is: world)

**eval:** Executes a command

**example:** cmd="ls -l"  
eval \$cmd

---

## **Examples of Bourne Shell Scripts**

**mantopdf:** converts a man page to pdf

```
#!/bin/sh

if test $# -le 0
then
    echo "usage mantopdf <file or cmd>"
    echo "e.g., mantopdf touch"
    echo "e.g., mantopdf pdftotext.1"
    echo "e.g., mantopdf /usr/man/man1b/touch.1b"
    exit
fi

if test -f $1
then
    filepath=$1
else
    filepath=`whereis $1 | tr ' ' '\n' |
grep man | head -1`

    if test -z "`echo $filepath`"
    then
        echo "man page for $1 is not found"
        exit
    fi
fi
```

```

filename=`path2file $filepath`

echo file path is:
ls -lt $filepath
echo file name is:
echo $filename

troff -man $filepath > /tmp/t1$$
echo ..done troff
cat /tmp/t1$$ | dpost > /tmp/t2$$
echo ..done dpost

ps2pdf /tmp/t2$$ $HOME/$filename.pdf
echo ..done ps2pdf
echo "the pdf file is:"
ls -lt $HOME/$filename.pdf

rm /tmp/t1$$ /tmp/t2$$
echo DONE

```

### **path2file:**

```

#!/bin/sh

IFS=/

set $1

eval filename=$`echo $#`
echo $filename

```

### **mantopdfs:** simple version of **mantopdf**

```

troff -man $filepath | dpost | ps2pdf -
$HOME/$filename.pdf

```

Replaces:

```

troff -man $filepath > /tmp/t1$$

```

```
cat /tmp/t1$$ | dpost > /tmp/t2$$  
ps2pdf /tmp/t2$$ $HOME/$filename.pdf
```

**Ex1:** (sh)

This program *mails a file to a group of users*.  
The first attribute is the file name, followed by the recipients' email.

```
#!/bin/sh  
echo `Usage : ex1 letter u1 u2 ....//send letter  
to users u1 u2 ..`  
letter=$1  
total=$#  
index=2  
while test $index -le $total  
do  
    eval addr=$`echo $index`  
    Mail $addr < $letter  
    echo Mailed $letter to $addr  
    index=`expr $index + 1`  
done
```

**Ex1:** (csh)

This program *mails a file to a group of users*.  
The first attribute is the file name, followed by the recipients' email.

```
#!/bin/csh  
echo `Usage : ex1 letter u1 u2 ....//send letter  
to users u1 u2 ..`  
set letter=$1  
set total=$#argv  
set index=2  
while ($index <= $total)  
    mail $argv[$index] < $letter  
    echo mailed $letter to $argv[$index]  
    @ index++  
end
```

### Ex2 :

This program informs you *when a specific user login* to your machine. It has one argument, the user to wait for.

```
#!/bin/sh

echo `Usage: ex2 user // wait until <usr>
login//`

until who | grep $1
do
    sleep 3
done
```

### Ex3 :

This programs gives a *list of users currently logged on*, sorted by their login time/user names

```
#!/bin/sh

Echo `Usage: ex3 //list who sorted by time of
login//`

who > /tmp/f$$

sortout < /tmp/f$$

rm /tmp/f$$
```

sortout

```

#!/bin/sh

while :

do
  if read line
  then
      set `echo $line`
      echo $5 $1 >> /tmp/t$$ /* $1
      $5 to sort by user names */
  else
      sort /tmp/t$$
      rm /tmp/t$$
      exit
  fi
done

```

#### Ex4 :

This program prints the *number of files in a subtree*.  
 The only argument is the root of the tree to search in.

```

#!/bin/sh

echo `Usage: ex4 path //count files under path
subtree//`

touch /tmp/t$$

find $1 -type f -exec /usr/bin/echo "1\c" >>
/tmp/t$$ \;

wc -c < /tmp/t$$

rm /tmp/t$$

```

## Ex5 :

This program *creates a backup directory* for C files

```
#!/bin/sh

echo `Usage: ex5 //copy c programs to
$HOME/backup directory//`

if test ! -d $HOME/backup
then
    echo $HOME/backup does not exist
    mkdir $HOME/backup
    echo ... $HOME/backup is created
fi

for i in *.c
do
    if test ! -f $HOME/backup/$i

    then
        echo $i is not in $HOME/backup
        cp $i $HOME/backup/$i
        echo copied $i to $HOME/backup/$i

    else
        echo comparing $i with $HOME/backup/$i
        if cmp -s $i $HOME/backup/$i
        then
            echo .... identical...
        else
            echo .... different...
            echo copying $i to $HOME/backup/$i
            cp $i $HOME/backup/$i
        fi
    fi
done
```

## Ex6:

This program *recursively scans the current directory* and prompts the users for a command to execute on each file.

```
#!/bin/sh
echo `Usage: ex6 [<path>] //scan directory at
<path> recursively//`

case $# in
  0) dir=. ;;
  1) dir=$1 ;;
esac
echo .... scanning directory $dir
```

```
cd $dir
```

```
for i in *
do
  if test -d $dir/$i
  then
    $0 $dir/$i
  else
    ls -l $i
    echo -n type any command:
    read command
    eval $command
  fi
done
```

---

[Examples of C Shell Scripts](#)

---

