


Programming the Shell


Shell Scripting

- Control Flow
 - Conditional Execution
- Branch
 - test** statement
 - string checks
 - numeric checks
 - file status checks
 - if** statement
 - case** statement
- Loop
 - for** statement
 - while** statement




1

SPA-SHELL PROGRAMMING-II 1042-1999




test Command - string checks

success true (0)  failure false (non-zero)

test "\$s1" true if string s1 is non-null
test -n "\$s1" true if length of string s1 is non-null
test -z "\$s1" true if length of string s1 is null
test "\$s1" = "\$s2" true if strings are equal
test "\$s1" != "\$s2" true if strings are unequal

2


SPA-SHELL PROGRAMMING-II 1042-1999



test Command - numeric checks


```
test $n1 -eq $n2 true if integers are equal
test $n1 -ne $n2 true if integers are unequal
test $n1 -gt $n2 true if integer n1 > n2
test $n1 -ge $n2 true if integer n1 >= n2
test $n1 -lt $n2 true if integer n1 < n2
test $n1 -le $n2 true if integer n1 <= n2
```

test Command - file status checks
test -option filename



3

SPA-SHELL PROGRAMMING-II 1042-1999



if then statement

FLOW CHART

```

graph TD
    Start(( )) --> Decision{if test expression}
    Decision -- true --> Then[then commands]
    Decision -- false --> Join(( ))
    Then --> Join
    Join --> End((fi))
  
```

SYNTAX

```

if test expression
then
  commands
fi
  
```

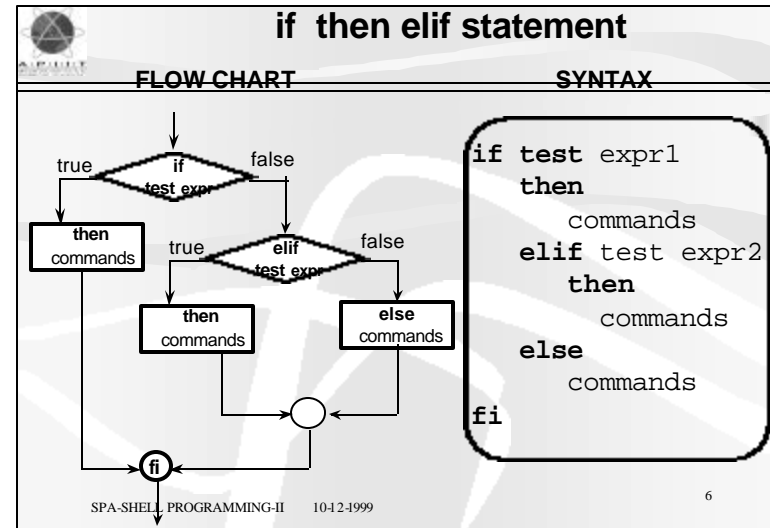
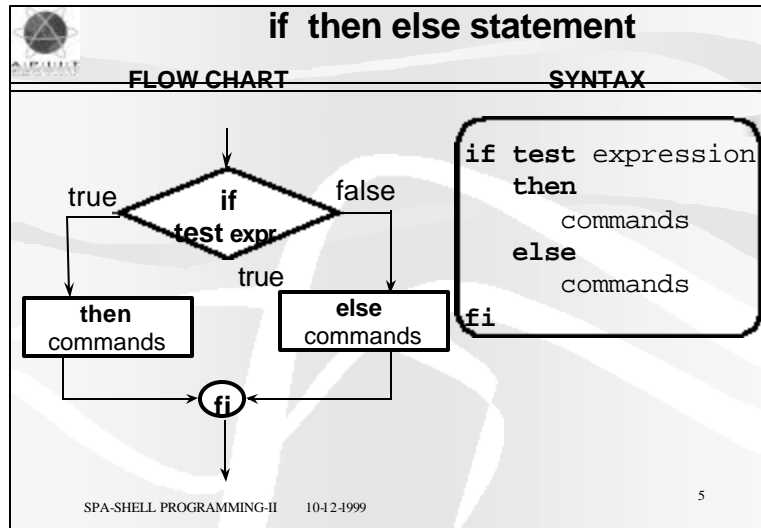
OR

```

if [expression]
then
  commands
fi
  
```

4

SPA-SHELL PROGRAMMING-II 1042-1999



Example -1

Write a bourne shell script to accept two names entered through the keyboard and check whether they are the same or not.

SPA-SHELL PROGRAMMING-II 1042-1999 7

Input from Keyboard: read statement

```

$ cat test1.sh
echo "Enter name1:\c"
read N1
echo "Enter name2:\c"
read N2
if test $N1 = $N2
then
  echo "name1 and name2 are equal"
else
  echo "name1 and name2 are not equal"
fi
  
```

SPA-SHELL PROGRAMMING-II 1042-1999 8



Example -2

Write a bourne shell script to accept a file name entered through the keyboard and checks whether it is a ordinary file or directory.

SPA-SHELL PROGRAMMING-II 1042-1999

9



Example-2

```
$ cat test11.sh
echo "enter filename: \c"
read FILE
if test -f $FILE
then
    echo "$FILE exists and is a regular file"
elif test -d "$FILE"
then
    echo "$FILE exists and is a directory"
else
    echo "$FILE does not exist OR \c"
    echo "is not a directory"
fi
```

SPA-SHELL PROGRAMMING-II 1042-1999

10



Example -3

Write a bourne shell script that requires to run with arguments.

SPA-SHELL PROGRAMMING-II 1042-1999

11



Example - 3

```
$ cat chkargs
if test $# = 0
then
    echo Usage : chkargs argument...
    exit 1
fi
echo Program running.
exit 0
$
```

SPA-SHELL PROGRAMMING-II 1042-1999

12

Example - 4

```

$ cat out
if [ $# = 0 ]
then
    echo "Usage: out [-v] filenames"
    exit 1
fi
if [ "$1" != "-v" ]
then
    cat "$@"
else
    shift
    more "$@"
fi

```

SPA-SHELL PROGRAMMING-II 1042-4999 13

Example - 5

```

$ cat if3
echo "word 1:\c"
read word1
echo "word 2:\c"
read word2
echo "word 3:\c"
read word3
if [ "$word1"="$word2" -a "$word2"="$word3" ]
then
    echo "Match: word 1, 2, & 3"
elif [ "$word1"="$word2" ]
then
    echo "Match: words 1 & 2"
elif [ "$word2"="$word3" ]
then
    echo "Match: word 2 & 3"
else
    echo "No match, Go Away!!"
fi
$

```

SPA-SHELL PROGRAMMING-II 1042-4999 14

case command

FLOW CHART

```

graph TD
    case((case)) --> d1{str = pattern-1?}
    d1 -- true --> c1[commands-1]
    d1 -- false --> d2{str = pattern-2?}
    d2 -- true --> c2[commands-2]
    d2 -- false --> d3{str = pattern-3?}
    d3 -- true --> c3[commands-3]
    d3 -- false --> dots[...]
    dots --> esac((esac))
    c1 --> esac
    c2 --> esac
    c3 --> esac
    
```

SYNTAX

```

case string in
    pattern-1)
        commands-1
        ;;
    pattern-2)
        commands-2
        ;;
    pattern-3)
        commands-3
        ;;
    .
    .
    .
esac

```

SPA-SHELL PROGRAMMING-II 1042-4999 15

Example of case command

```

$ cat catel
echo "Enter a character : \c"
read CH
case $CH in
    a) echo "You entered an 'a' "
        ;;
    b) echo "You entered a 'b' "
        ;;
    *) echo "You entered neither an \c"
        echo " 'a' nor a 'b' "
        ;;
esac

```

SPA-SHELL PROGRAMMING-II 1042-4999 16

Example of case command

```
$ cat cate2
echo "enter a character : \c"
read CH
case $CH in
  a|e|i|o|u) echo "vowel"
              ;;
  0|1|2|3|5) echo "0-5 entered"
              ;;
  *) echo "enter vowel or number 0-5"
      ;;
esac
```

SPA-SHELL PROGRAMMING-II 1042-1999 17

Arithmetic expressions

- Utility: **expr** expression
 - Evaluate an expression
 - Components of expression must be separated by blanks
- Operators (Note: Use \ (and \) for brackets)

*	/	%	Multiplication, division, remainder			
+	-		Addition, subtraction			
=	>	>=	<	<=	!=	Comparison operators
&						Logical AND
						Logical OR

SPA-SHELL PROGRAMMING-II 1042-1999 18

String operators

substr string start length
 - Returns substring of string

index string charList
 - Returns the index of the first character in string which appears in charList

length string
 - Returns the length of the string

SPA-SHELL PROGRAMMING-II 1042-1999 19

Expression example

```
$ x=1
$ x=`expr $x + 1`
$ echo $x
2
$ echo `expr substr "donkey" 4 3`
key
```

```
#!/bin/sh
echo `expr $1 + $2`
```

```
$ add 10 12
22
```

SPA-SHELL PROGRAMMING-II 1042-1999 20

The while LOOP

- Execute a set of statements *while* a condition remains *true*. Exit when condition becomes *false*.

Format

```

$ cat wil.sh
read VAR
while command1
do
    command(s)
done
    while test -n "$VAR"
    do
        echo $VAR >> file
        read VAR
    done
    cat file
  
```

21

SPA-SHELL PROGRAMMING-II 1042-1999

The until LOOP

- Execute a set of statements *until* a condition remains *false*. Exit when condition becomes *true*.

Format

```

$ cat until.sh
read VAR
until command1
do
    command(s)
done
    until test -z "$VAR"
    do
        echo $VAR >> file
        read VAR
    done
    cat file
  
```


22

SPA-SHELL PROGRAMMING-II 1042-1999

Exercise - 1 :What's happening here ?

```

$ puz1.sh 3
?
$ puz1.sh
10
?
  
```



puz1.sh

```

#!/bin/sh
x=1
while [ $x -le $1 ]
do
    echo "`expr $x \* 2` \c"
    x=`expr $x + 1`
done
echo
  
```

23

SPA-SHELL PROGRAMMING-II 1042-1999

The for LOOP

- Execute set of statements certain number of times.

Format

```


for i in arg-list
do
    command(s)
done
  
```

```

$ cat for1.sh
for i in 1 2 3 4 5
do
    echo "$i \c"
done
  
```

24

SPA-SHELL PROGRAMMING-II 1042-1999




Example of for usage

- Other ways of generating 'argument list'
e.g.,

```
$ cat dir
for1
for2
for3
$ cat for3
for i in `cat dir`
do
    grep 'for' $i
done
$
```

SPA-SHELL PROGRAMMING-II 1042-1999 25




The for LOOP - 2nd form

```
for loop-index
do
    commands
done
```

Example:

```
$ cat for_test
for args
do
    echo $args
done
$ for_test good bad ugly
good
bad
ugly
$
```


SPA-SHELL PROGRAMMING-II 1042-1999 26



A Sample User Identity Script

```
$ cat whoon
:
if [ $# = 0 ]
then
    echo "Usage: whos id..."
    exit 1
fi
for i
do
    awk -F: '{print $1, $5}' /etc/passwd |
    grep -i "$i"
done
```

SPA-SHELL PROGRAMMING-II 1042-1999 27



Partial Personalities at APIIT !

```
$ whoon con

hf960918 Constance Ng S.C,,,
hf960973 SEAN CONNERY,APIIT,,
mu970303 FunnyName,Thailand,Confidential,Confidential
df980303 Top Secret!!,Subang,Confidential,Confidential
mu980311 CoCONUT,,,
df981147 coconut,apiit damansara,,
```

SPA-SHELL PROGRAMMING-II 1042-1999 28

Functions

Format

```
function_name()
{
    commands
}
$ whoson ()
{
    date
    echo Users Currently Logged In
    who
}
```

SPA-SHELL PROGRAMMING-II 1042-1999 29

Setting shell options

Use set in a shell script to replace the positional command line arguments in \$1,\$2,... by the set arguments

```
$ set how are you
$ echo $1
how
$ echo $2
are
$ echo $3
you
```

SPA-SHELL PROGRAMMING-II 1042-1999 30

Job Control

Commands to control your processes
Monitor the status of processes
Utility: ps -arux
By default, ps lists information about processes created by your current shell.

- a include processes owned by all users
- x include also processes that do not have a terminal
- r list only processes that are running
- u user-oriented output including each process' owner

SPA-SHELL PROGRAMMING-II 1042-1999 31

Example

```
$ c89 card1.e &
[1] 20870
$ ps
```

PID	TTY	TIME	COMMAND
20872	ttyp5	0:00	ps
20870	ttyp5	0:00	c89
20600	ttyp5	0:00	sh
20599	ttyp5	0:00	telnetd

It displays Process-Id, Terminal, Process State, Current CPU time used and the command name

SPA-SHELL PROGRAMMING-II 1042-1999 32



shift command

Handy in shell scripts

Shift causes all positional parameters $\$2 \dots \n to be renamed $\$1 \dots \$(n-1)$.

The original first parameter is lost.

runShift

```
#!/bin/sh
echo first arg is $1, all args are $*
shift
echo first arg is $1, all args are $*
```

```
$ runShift a b c d
```

```
first arg is a, all args are a b c d
```

```
first arg is b, all args are b c d
```

SPA-SHELL PROGRAMMING-II 10424999

33