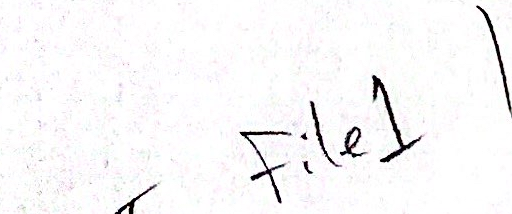
**10**

**Conditional Constructs**

**Objectives**

**Create conditional constructs using** if, if-else, **and** if-elif

**Create a conditional construct using** case





Notes

**Bourne Shell Programming**

· Programs or procedures can be written using Shell utilities, existing Shell programs, and Shell programming constructs. By combining these features,it is possible to:

Construct a Shell command file which can be executed as any Shell command or utility;

Pass parameters to the command file from the command line;

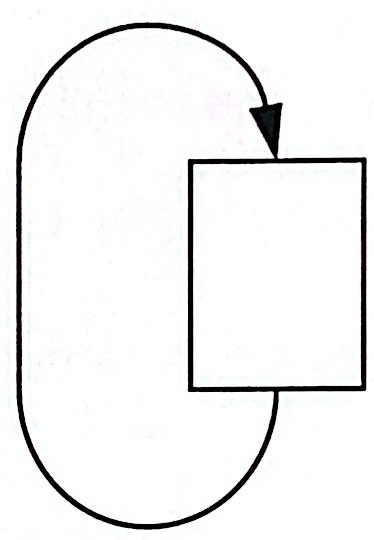
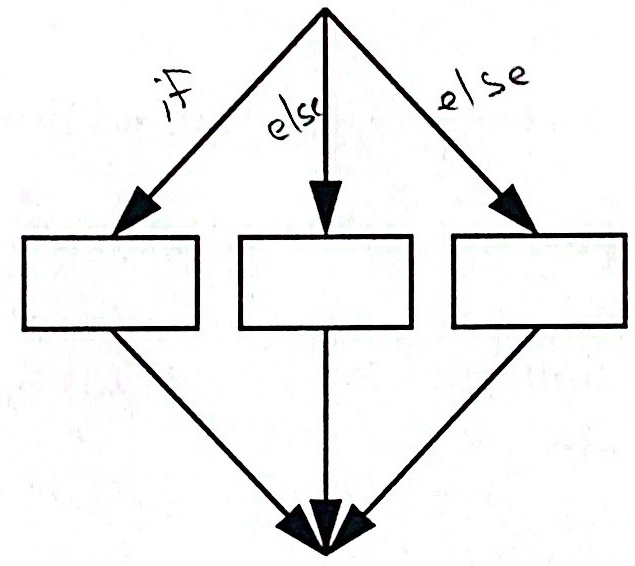
Define variables within a Shell program; and

Alter the flow of execution of the command file by using the control structures if,for, case, while and until.

Notes

**Programming Concepts**

Branches Loops





**Notes**

· All programming flow control falls into two broad categories: branching and looping. In this chapter we discuss branching, or conditional constructs.

Good programming practice: initialize variables before using.

**Advantages/Disadvantages of Shell Scripts**

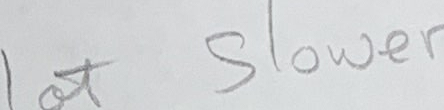
Advantages:

1. Easy to change

2.Don't have to be re-complied, re-linked, and re-loaded when changes are made

3. Easy to debug

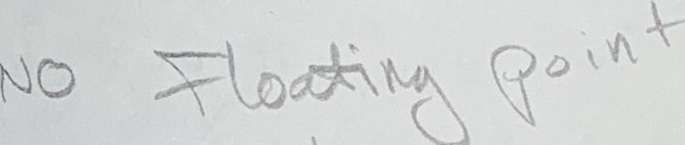
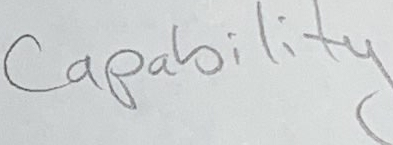
4. Provide a good introduction to programming

Disadvantages:

1. Slower than compiled programs

2. Do not have type checking

3. Have modest floating point capability

**Notes**

Shell files are executed in much the same way as a BASIC program. Each line is interpreted and executed.

**Flow Control-if**

**if** test expression 

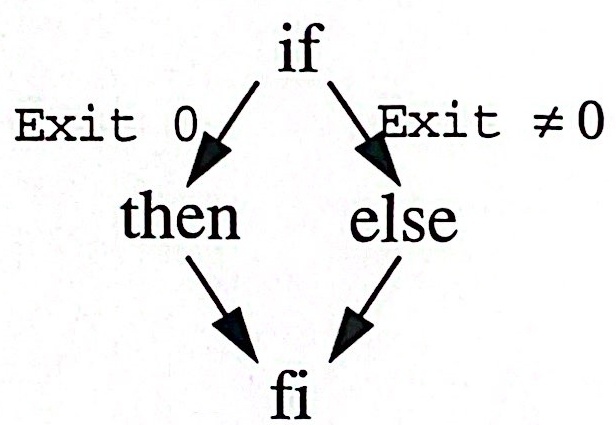
**then**

command-list1

**e1se**

command-list2

**fi**



**Notes**

The i f statement is a two-way conditional branch.

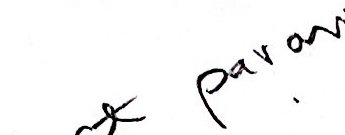
The value of the expression following i f is tested.

If it is true, then the command-list following then up to else is executed.

If it is false, then the command-list following else up to fi is executed (a true value is the value zero, non-zero is not true).

Each of the keywords must be the first word on a line.

**Flow Control-if Examples**



**Example 1:**

**if** test -f

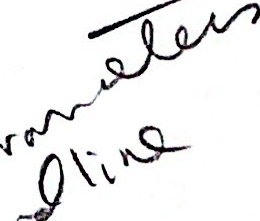
**then**

cat

**e1se**

1s -1 $1

fi



**Example 2:**

**if** test("$#"-eq 0

**then**

echo "a filename is missing"

**else**

sort $\*

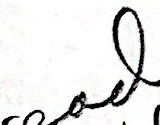
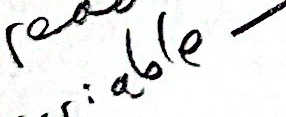
fi

**Example 3:**

echo "enter first word"

read word1

echo "enter second word"

read word2

**if** test "$word1" = "$word2"

**then** echo "You entered the same word"

**fi**

**Notes**

· The equal sign used in the test statement must have spaces surrounding it.

· The else clause is optional.



**Flow Control - if Examples (Cont'd)**

**·Example 4:**



max=5

users=`who

**if** test "Susers"-gt "$max"

**then**

echo "the system is busy"; exit

**else**

echo "you may login"

fi

**· Example 5:**

**if** test "$#" -ne 1

**then**

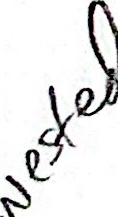
echo "Argument missing"

echo "Usage: whois name"

exit 2

**else**

user=$1

 **then** if who/dev/nul1

echo"Suser is logged on"

**else**

echo "$user is not logged on"

**fi**

**Notes**

· Two fi's are required in Example 5.

**Flow Control-elif**

**· elif is an implementation of a multi-way branch.**

**if** test expression1

**then** command-list1

**elif** test expression2

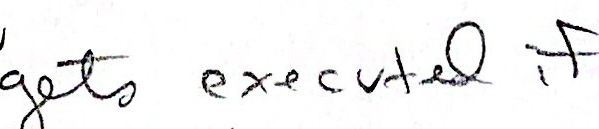
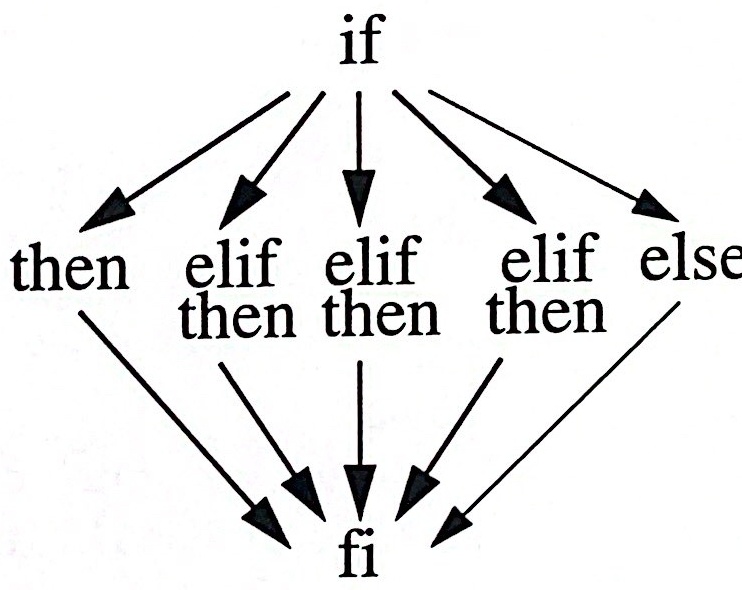
**then** command-list2

**elif** test expressionk

**then** command-listk

**else** command-listk+1

**fi**

**Notes**

This structure allows you to construct a nested set of if-then-else controls.

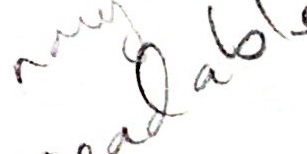
Any expression found to be true passes control to the next then where the command list is executed.

Immediately following the execution of the commandistcontrolispassedcthestatementfollowing fi.

Care must be exercised to assure that the test expressions are mutually exclusive.

**Flow Control - elif Examples**

**· Example 1:**

**if** test -z "$1"

**then**

echo "You did/not enter an argument"

**elif** test -f $1 -a -r $1

**then**

cat $1

**elif** test -d $1

**then**

echo "the file is a directory"

**else** echo "$1 is unknown"

**fi**

**Example 2:**

**if** test -f $1 

**then**



**elif** test

**then**

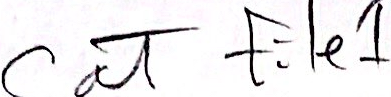
(cd $1;pwd)

**else**

echo $1 is a special file

fi 

**Notes**



Conditional Constructs

**Flow Control- case**

**case** string **in**

pattern-1)

command-list1;;

pattern-2 )

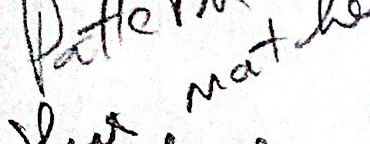
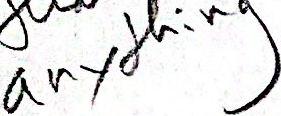
command-list2;;

pattern-k)

command-listk;;

\* )default-command-list

command-listk+1;;

**esac** 



**Notes**

The Shell attempts to match string with each pattern in the order in which the patterns appear.

If a match is found, the command-list is executed and the case is finished.

case does not check for duplicate patterns. The first matching pattern is used.

If none of the labels match before a default label is found, the default command list is executed,and the execution resumes after the esac.

The case statement is a multi-way branch.



**Flow Control - case Example**



**·Example:**

**case**

echo "No file was entered";;

1)

echo "Enter information followed by CTRL-d"

cat>> $1;;

2)

cat >> $2< $1

cat $2;;

\*)

echo "Too many file names.";;

**esac**



**Notes**

**Patterns in the case Structure**

**The Shell tries to match the** string **with each pattern. Once a pattern is** **matched and its command list executed, control falls through the** case **structure to the line following esac. If the string matches several** patterns,**only the command-list after the first match is executed. After** execution,**control falls through to the end of the** case **structure.**

\* Matches any string of characters

? Matches a single character

[...] Defines a character class

[a-z] Matches a range of characters

- Separates alternate choices

**Notes**

Since \* matches any string, it can be used as a default pattern.The characters shown above can be used in patterns.

It is necessary to quote the above special characters with a backslash when they are used explicitly in the pattern.

**case Examples**

**· Example 1:**

**Explain the difference in the following two examples.**

**a. case** $1 **in**

**1)** echo "you entered a 1";;

2) echo "you entered a 2";;

\*) echo "neither a 1 nor a 2";;

**esac**

**b. case** $1 **in**

1|2) echo "you entered a 1 or a 2";;\*echo "neither a 1 nor a 2";;

**esac**

**Example 2:**

echo "Enter an integer between 0 and 9"read integer

**case** $integer **in**

[1-5])echo "You entered an integer

between 1 and 5";;

[6-9]|0) echo "You entered an integer

between 6 and 9 or 0";;

\*) echo "You did not follow directions";;

**esac**

**Notes**

UNIX Shell Programming 10-14

©Copyright 1996-2000 GTRC, All Rights Reserved

**case Examples(Cont'd)**

**·Example 3:**

# Display character type

**if** test "$#" -ne 1

**then**

echo "Usage: chartype char"

exit 1

**fi**

char="$1"

numchars=`echo "$char" | wc -c`

**if** test "$numchars" -ne 2 # character followed

**then** # by new line

echo "Please type a single character"

exit 1

**fi**

**case** "$char" **in**

[0-9]) echo "Digit";;

[a-z]|[A-Z]) echo "Letter";;

?) echo "Special character";;

\*) echo "Please make another

selection.";;

**esac**

**Notes**

**Summary**

·if **...** then **...** fiis **the principle conditional construct in shell** **programming.**

Use if ... then ... elif ... then ...to **have** **multiple** **related** tests.

**An** else **can be used at the end for the "default" case.**

Thecase...esac **statement can be used to match text against different** shellpatterns. Itis **often used when processing user responses.**



**Notes**