11

**Looping Constructs**

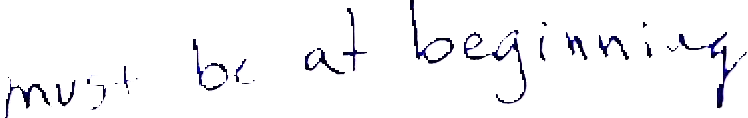
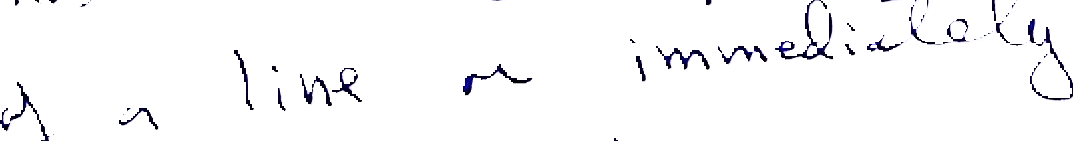
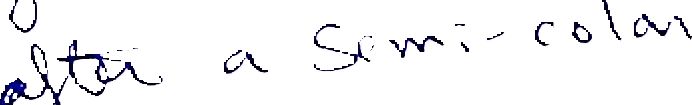
**Objectives**

**·Use the looping constructs** for, while **and** until

**·Use the break and** cont inue **statements to prematurely exit from a loop**

Notes





**Notes**

· A command-list is a sequence of simple commands separated by a semicolon or a new line.

Keywords do and done are only recognized following a semicolon or a new line.

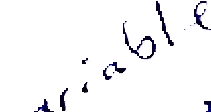
·When the loop is executed, the first string string1 is assigned to the variable and the command-list is executed. Then string2 is assigned to the variable and the command-list is executed again. The process continuesuntil the last string stringn is assigned to the variable.

If there are n strings in the list, the command-list is executed n times.

In this chapter, we begin the other broad category of flow control: looping. All looping constructshavean **initialization, a test for completion,** and **an incrementing** orameansof progressing through the loop. Any one or more of these three parts may be implicit.



**Flow Control- for in Examples**

 **· Example 1:** 

**for** name in bill joe mary

**do**

echo $name

**done**

**Example 2:**

**for** arg **in** $\*

**do**

echo $arg

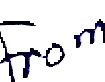
**done**



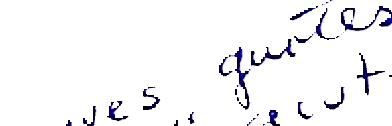
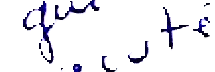
**Example 3:**

**for** memo in "memo.?"

**do**

  mail rice < "$memo"

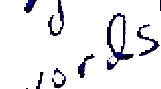
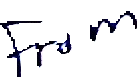
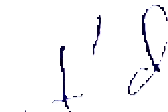
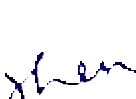
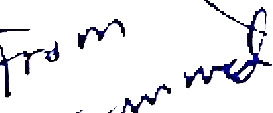
done

Example 4:

for i in '1s \*

do

      Notes

done

Looping Constructs

**Flow Control - while**

**while** expression

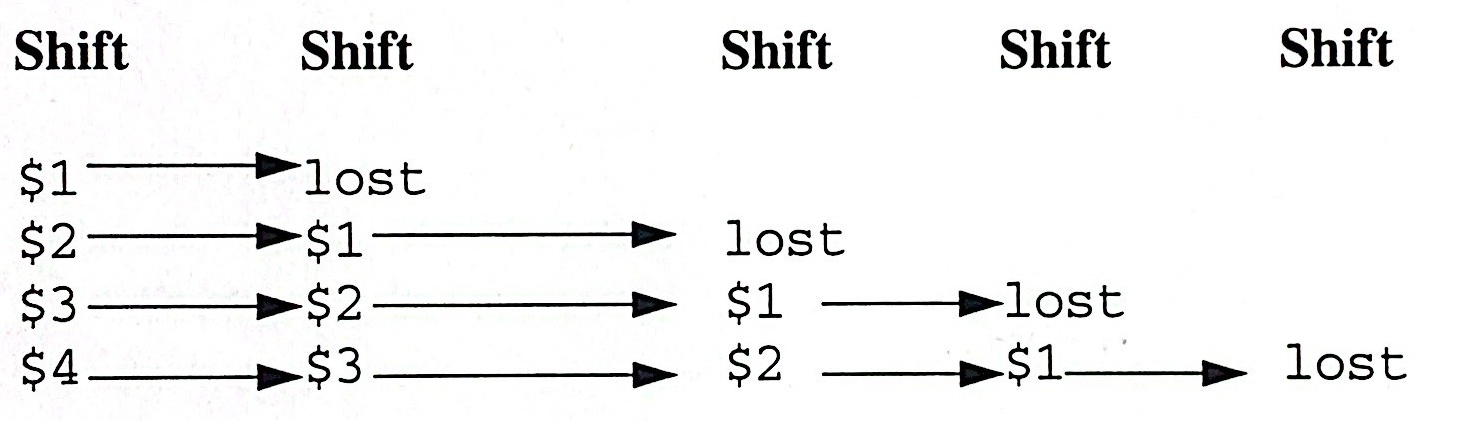
**do**

command-list

**done**

shift

**The shi ft statement is very useful in a while statement.It renames** **the positional parameters as represented in the following table.**



shiftn **causes n consecutive shifts.**

**Notes**

The value tested by the whi le command is the exit value of the last simple statement in the expression following the keyword while. If a zero value is returned,then the command-list between do and done is executed.

When the exit status of the expression is false, control passes to the statement following done.

The command-list will never be executed if the expression returns a nonzero exit status the first time it is evaluated.

The initiaization of the while and reinitialization can be done using the shift or an explicit counter.

The shi ft refers to the position parameters, the user-defined counter should be initialized before the while loop is entered, tested and incremented or decremented within the while.

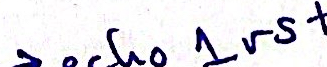
The Shell retains all positional parameters, but you can only access $0 through $9.

**Flow Control -while Example**

**·Example 1:**

**while** x$#-ne 0

**do**

echo $1

**shift**

**done**

**Example 2:**

count=$#

**if** test $# -eq 0

**then**

echo "You did not enter a file"

exit 1

**fi**

**while** test $count -gt 0

**do** 

**if** test -f $1

**then**

cat $1

1s -1 $1

**else**

echo "$1 is a directory"

**fi**

count=`expr $count-1`

**shift**

**done**

**Example 3:**

count=0

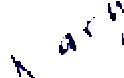
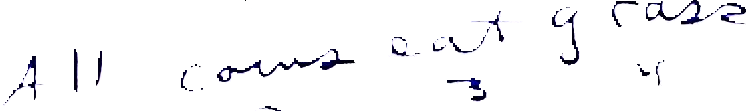
**while** test $count -1t 10

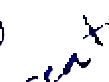
**do**

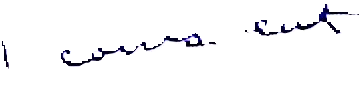
count=`expr $count+1

echo $count

**done**

**Flow Control - while Example**  

**Example 4:** 

**while** test $# -gt 0  echo $\*→

**do**

echo $#

**shift**

**done**

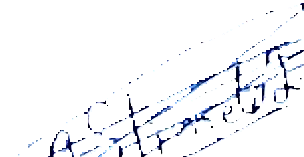
shift **renumbers the arguments and decrements $#.**

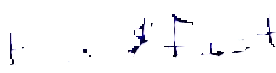
**Example 5:** **Factorial**

**Usage:** factorial **number**

fact=1

count=$1

**while** test $count -gt **do** 

expr $count  

count=`expr $count 

**done**

echo $fact

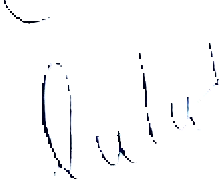
**Example 6:** **Exponentiation**

**Usage: number base power**



base=$1

power=$2

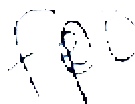
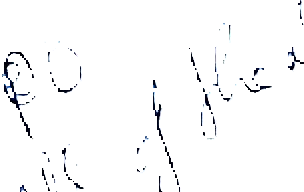


result=1

**while** test $power -gt 0

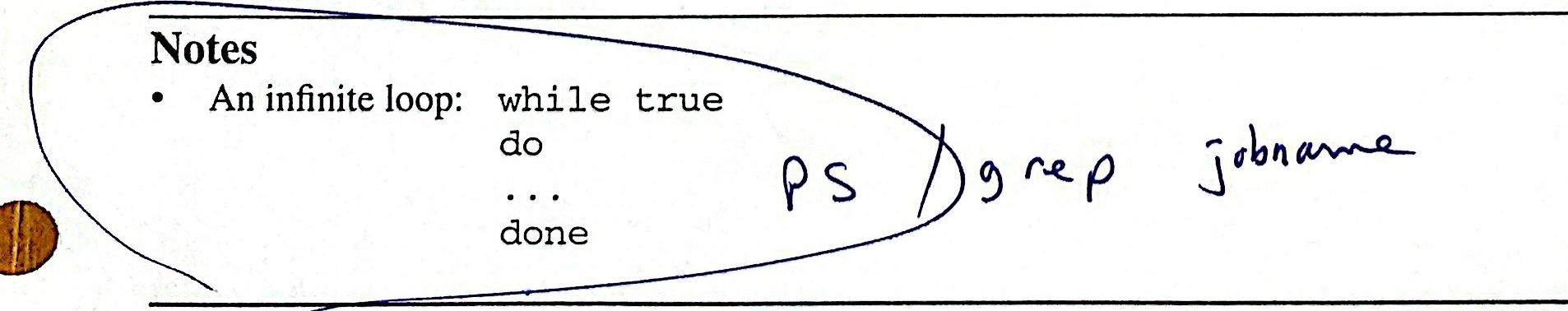
**do**

result=`expr $result \\* $base`

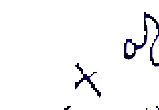
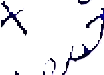
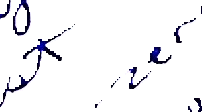
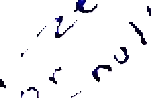
power=`expr $power

**done**

echo $result 



**Comparison Between for and while**

**while**

**while** test -n "$1"

**do**

shift

**done**

**for**

**for** i

**do**

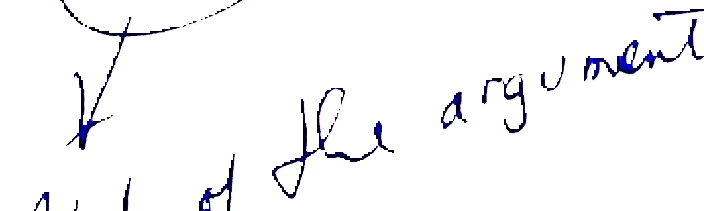
**done**



**Notes**

Create an example to illustrate these two concepts.

for i is equivalent to for i in "$@".



**for,while Examples**

**·Example 1:**

$ set abc

$ echo $1 $2 $3

abc

$ **for** var; do echo $var; **done**

a

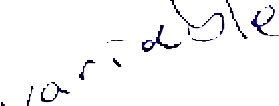
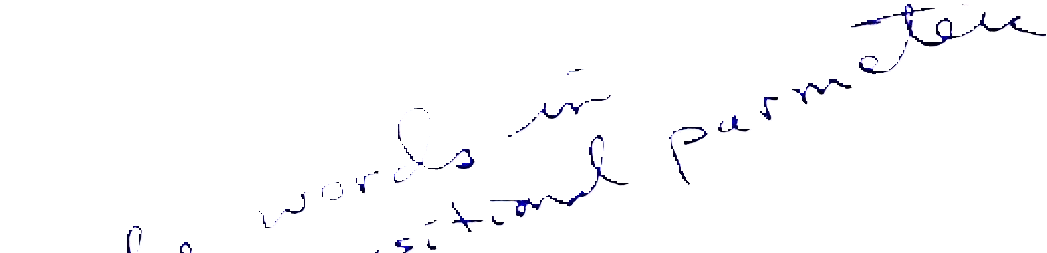
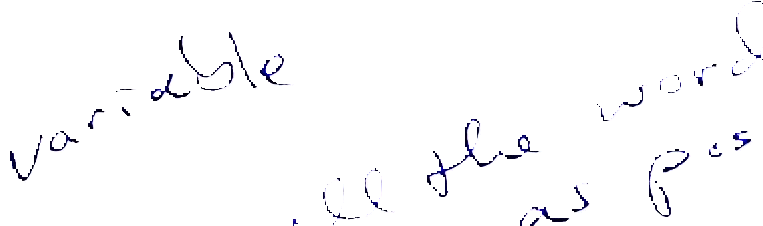
b

C

$

**The positional parameters cannot be directly assigned values.** Remember **the** set **command along with arguments assigns values to the positional** **parameters.**

**Example 2:**

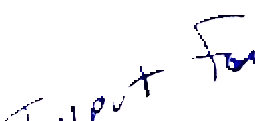
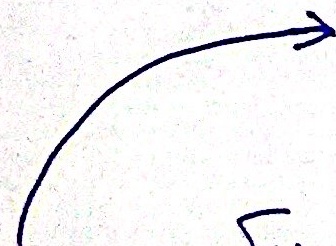
$ cat words

count=0

**while** read line

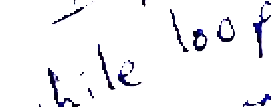
**do**

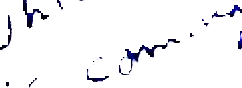
set $line

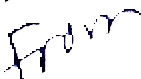
 count=`expr $count +$#

**done** < filename

echo $count



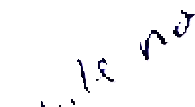
 **Counts the number of words in a file.**

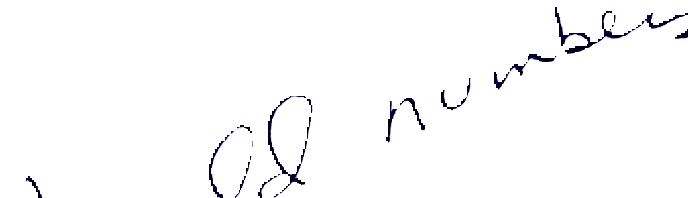
 **Notes**



UNIX Shell Programming 11-9

©Copyright 1996-2000 GTRC,All Rights Reserved

 **Flow Control - until**

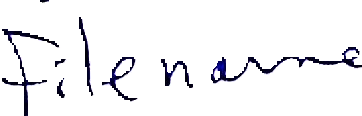
**until** expression 

**do**

command-list

**done**



**Notes**

unti1 inverts the test of loop termination. whi le continues looping until the test returns a false value. until terminates when a true value is returned.

until is useful in writing programs that require a wait period until a certain event occurs.



**until Examples**

**·Example 1:**

**if** test $# -eq 0

**then** echo "Usage: mon user"; exit 1

**fi**

user=$1

**until** who|grep

**do**

sleep 60

**done**

echo "$user has logged on."

**· Example 2:**

password=hi

name=guess

echo "What is the password?"

**until** test "$password" = "$name"

**do**

echo "\n Password is: \c"

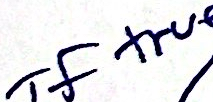
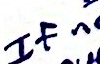
read name

**done**

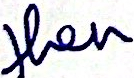
echo "You got it!!!"

**Example 3:Check Mail**

**until** test -f /usr/spoo1/mail/$1 #may be located

  #in /usr/mail/$1 sleep 300**do**  

**done**

 echo "you have mail"

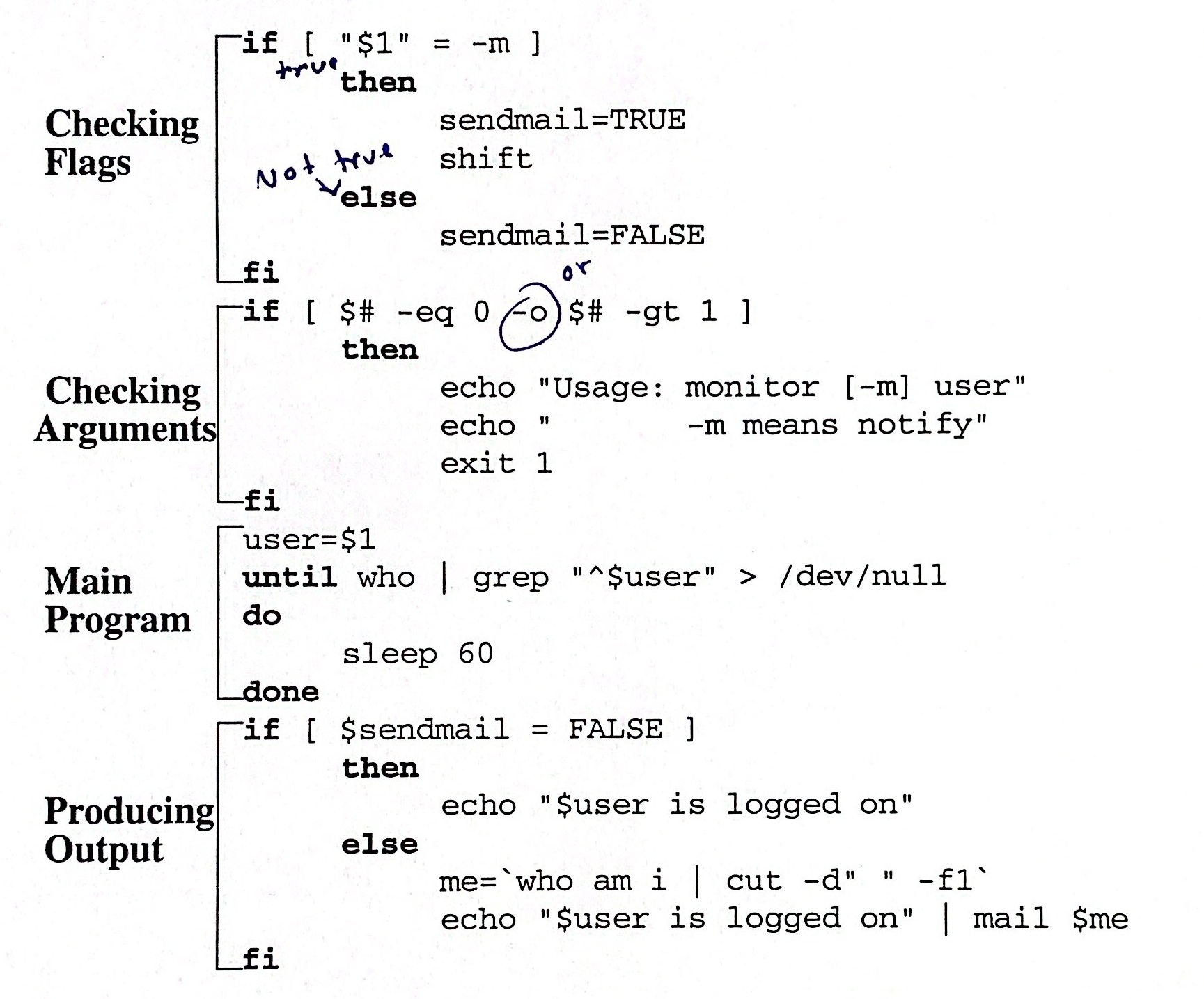
**Notes**

In example 1, the loop continues until grep is successful.



**until Examples(Cont'd)**

**· Example 4: Monitor Users**





**Notes**

UNIX Shell Programming 11-12

© Copyright 1996-2000 GTRC, All Rights Reserved

**Programming Notes**

**1. Dispose of system or command messages by directing output to** /dev/null.

who | grep tom > /dev/nu11

**2. Match exact patterns by anchoring the pattern using ^ for the** beginningof **a line or $ for the end of a line. The exact pattern should be enclosed in** **double quotes and appropriate spaces should be inserted.**

who | grep "^tom"

**matches only the name tom at the beginning of a line.**

**3. Use the null command “:”when you want the system to do nothing. If** afile **exists,then do nothing. However, if it does not exist, you may want to** echo **an error message.**

if [ -f data ]

then

:

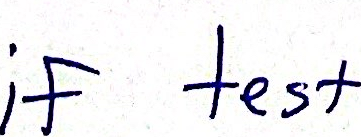
else

echo "data is missing"

exit 1

fi

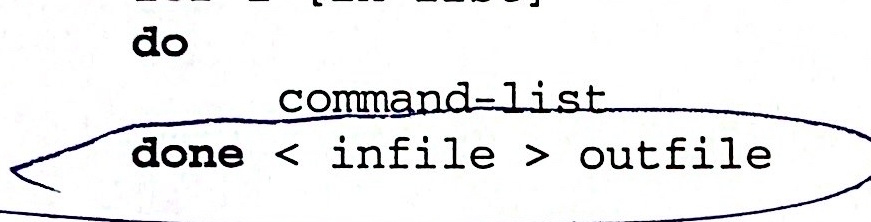
**Notes**



**Programming Notes (Cont'd)**

**4. Redirect output of a loop.**

**for** i [in list]



**Redirect standard error by:** done>outfile2>errorfile

5.forname **implies for** namein"$@" **where "$@" expands into the** **entire list of arguments on the command line.**

**6.** **Using** break **and** continue.

break **and** continue **are used to alter execution of the program within a** **loop.**

**while** true

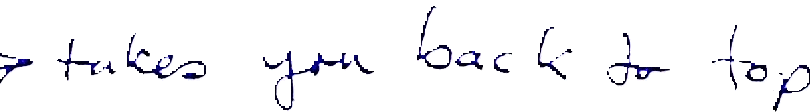
**do**

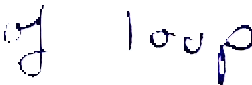
read x

**case** $x in

"exit") echo you are through

**break;;**

"") echo enter more data **continue;;**

\*) echo you entered $x;; 

**esac**

**done**

**Notes**

continue causes control to resume at the beginning of the next loop.

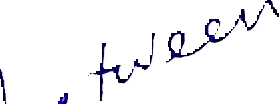
break terminates execution of the entire iteration on the innermost loop. It is possible to break from several levels.

**break** n

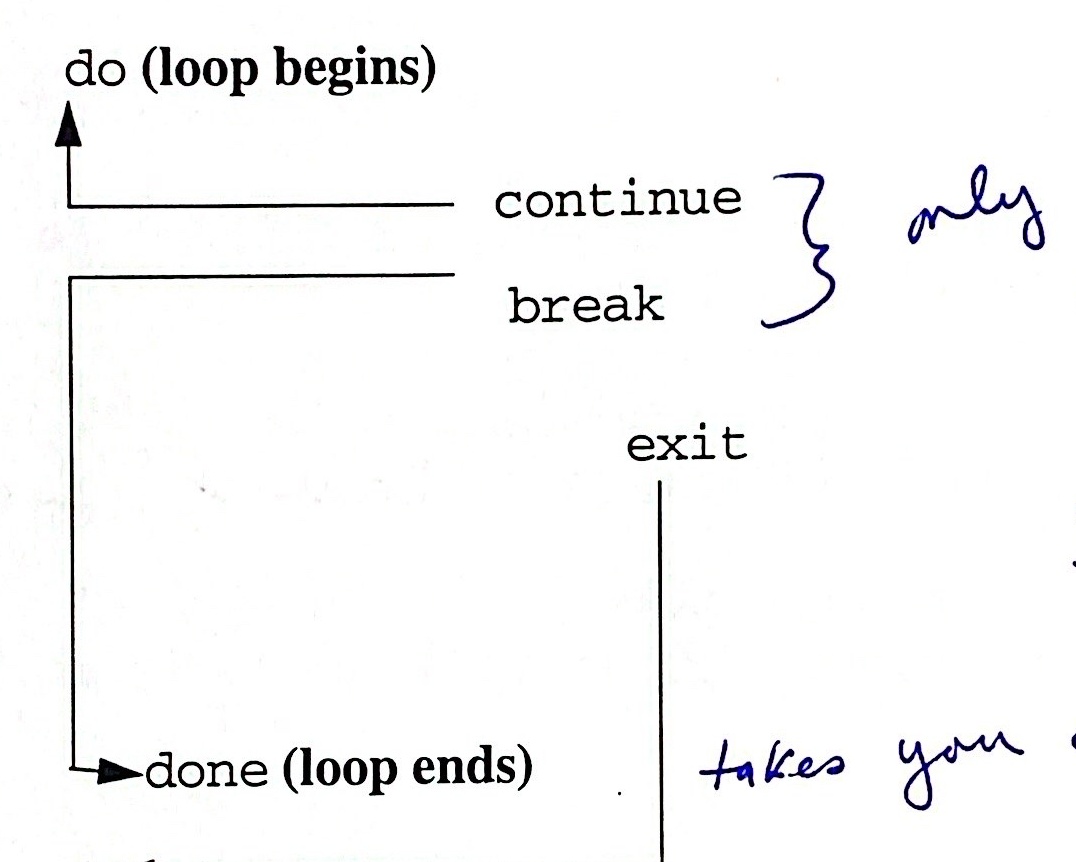
Exits n levels.



**Programming Notes (Cont'd)**





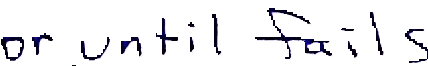
**Notes**

break allows exting from the loop and not from the program. break is used to terminate a loop when an error condition is detected.

break can be used to terminate execution of a for loop and continue can be used to prematurely terminate one iteration and begin the next. break and continue must be placed between the keywords do and done.

**Summary**

·The shell has three **looping** **constructs,** while, until,and for.



·A while loop runs until the test command exits non-zero.

· An unti1 loop runs until the test command exits zero.

The for loop loops a variable over a set of values. The default set of values are the positional parameters.

Use break to break out of a loop. An optionalnumeric argument breaks out of multiple enclosing loops. 

Use cont inue to skip the rest of the loop body and start the next iteration.An optional numeric argument continues at a higher enclosing loop.



Notes