Shell Programming II

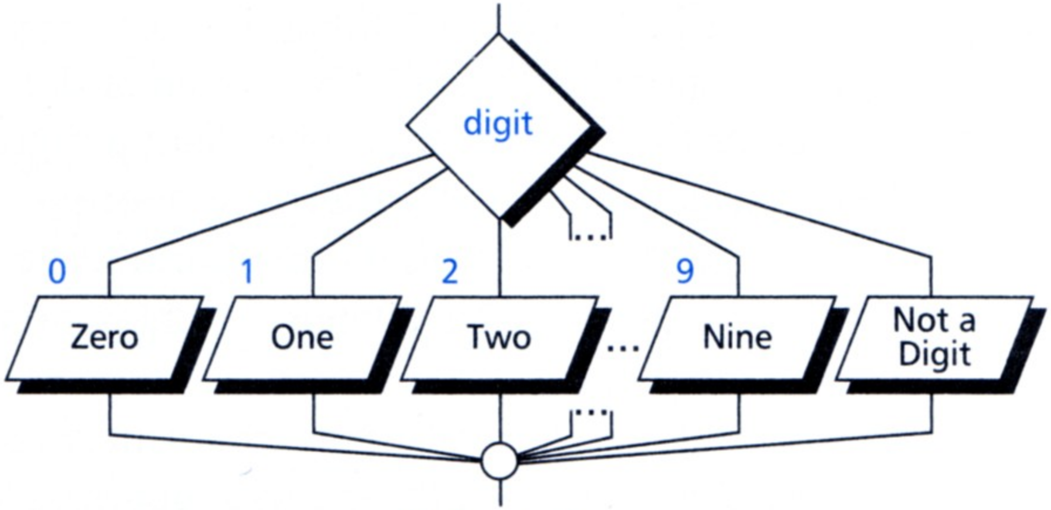
01418235

Chavalit Srisathapornphat

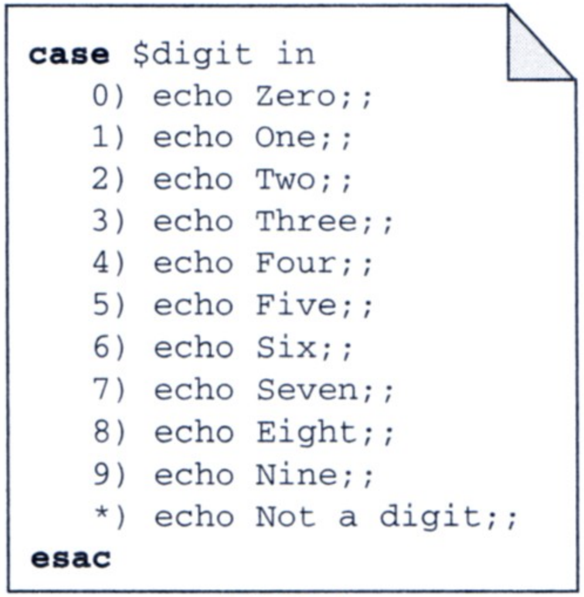
Outline

- case statement
- Loops
  - while
  - until
  - for … in
  - select
- Command line parameters

case



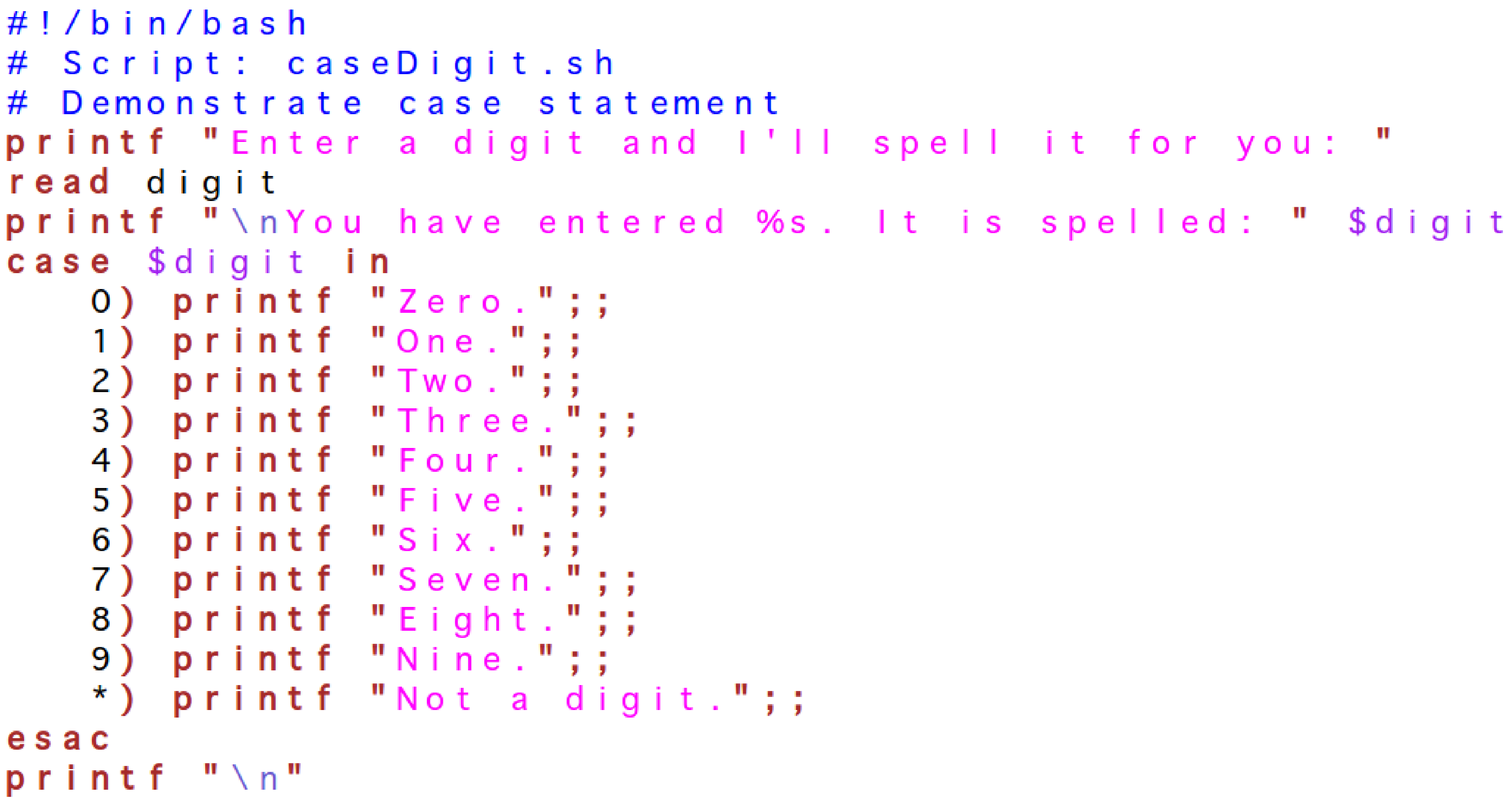
(a) Logic Flow



```
case $digit in
    0) echo Zero;;
    1) echo One;;
    2) echo Two;;
    3) echo Three;;
    4) echo Four;;
    5) echo Five;;
    6) echo Six;;
    7) echo Seven;;
    8) echo Eight;;
    9) echo Nine;;
    *) echo Not a digit;;
esac
```

(b) Code

case: Example I



```
#!/bin/bash
# Script: caseDigit.sh
# Demonstrate case statement
printf "Enter a digit and I'll spell it for you: "
read digit
printf "\nYou have entered %s. It is spelled: " $digit
case $digit in
    0) printf "Zero.";;
    1) printf "One.";;
    2) printf "Two.";;
    3) printf "Three.";;
    4) printf "Four.";;
    5) printf "Five.";;
    6) printf "Six.";;
    7) printf "Seven.";;
    8) printf "Eight.";;
    9) printf "Nine.";;
    *) printf "Not a digit.";;
esac
printf "\n"
```

case: Example II

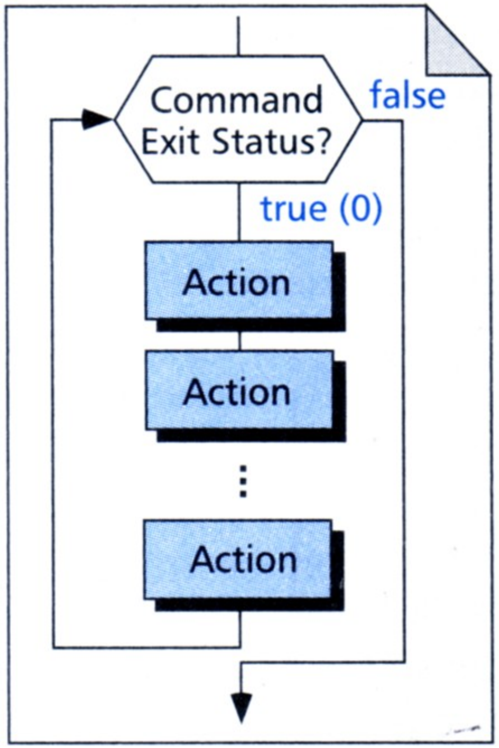
```
hour=$(date|cut -c 12-16)

case $hour in

  0?:??|1[01]:??) printf "Good morning. It's %s A.M." $hour ;;

  1[2-7]:??)      printf "Good afternoon. It's %s P.M." $hour ;;

  1[89]:??|2?:??) printf "Good evening. It's %s P.M." $hour ;;

  *)              printf "Sorry, I don't know the time" ;;

esac
```

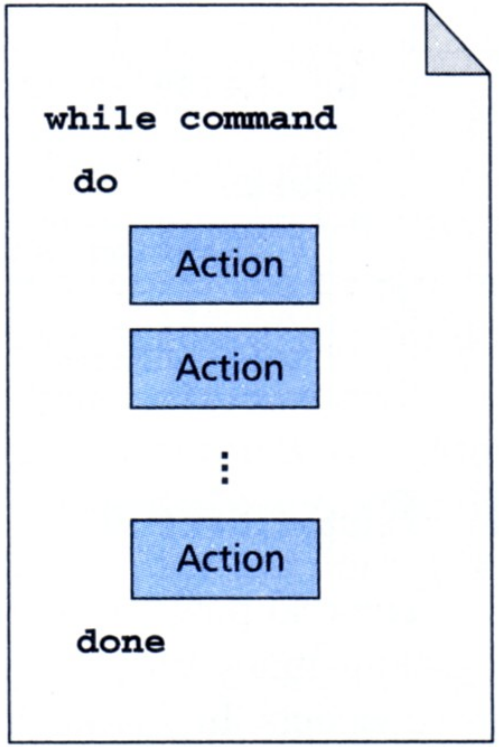
- Redo this script with "case" statement
  1. Write a shell script that checks if it is a winter (Nov-Feb), summer (Mar-Jun), or rainy (Jul-Oct) season.

Loops

- Command-controlled loops
  - □ while
  - □ until
- List-controlled loops
  - for ... in
  - select
- Arithmetic for loops
  - □ for ((init ; ending_condition ; update))

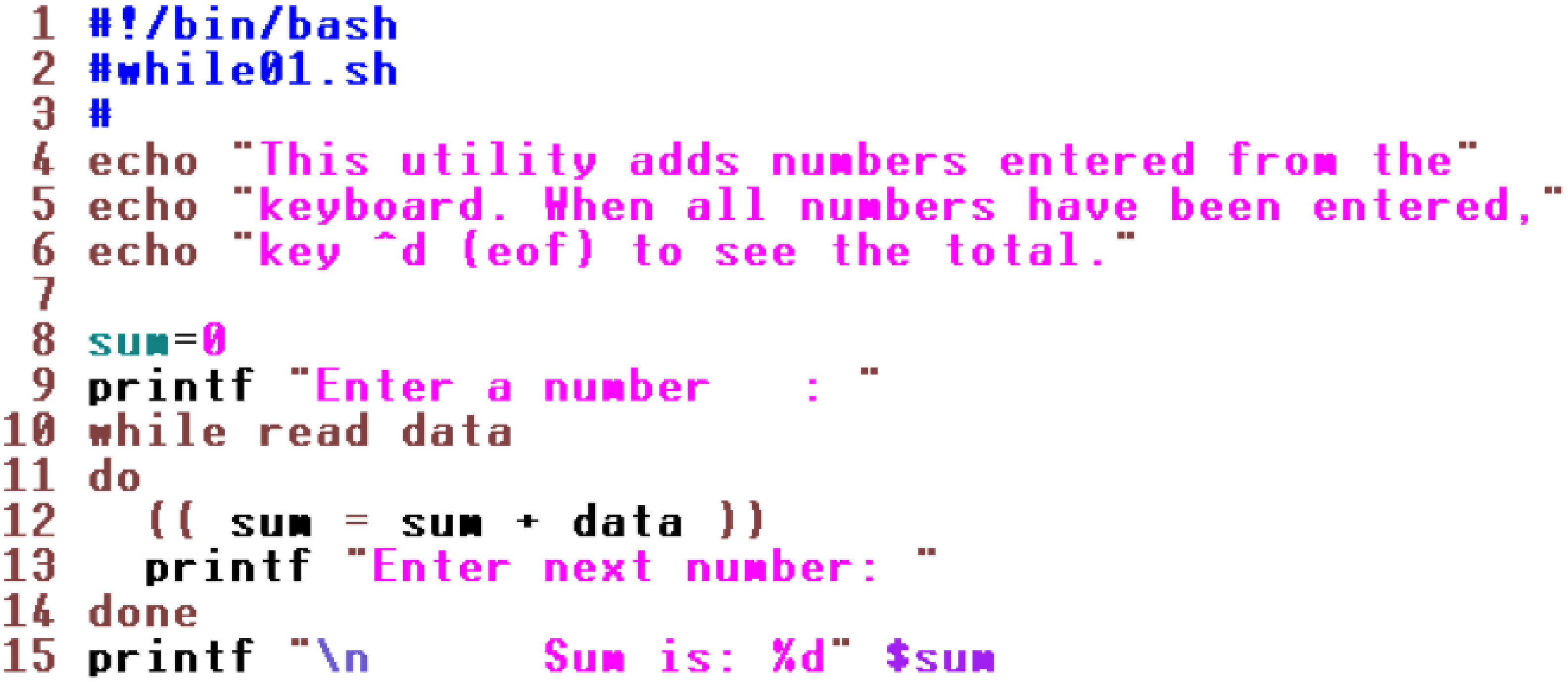
while loop





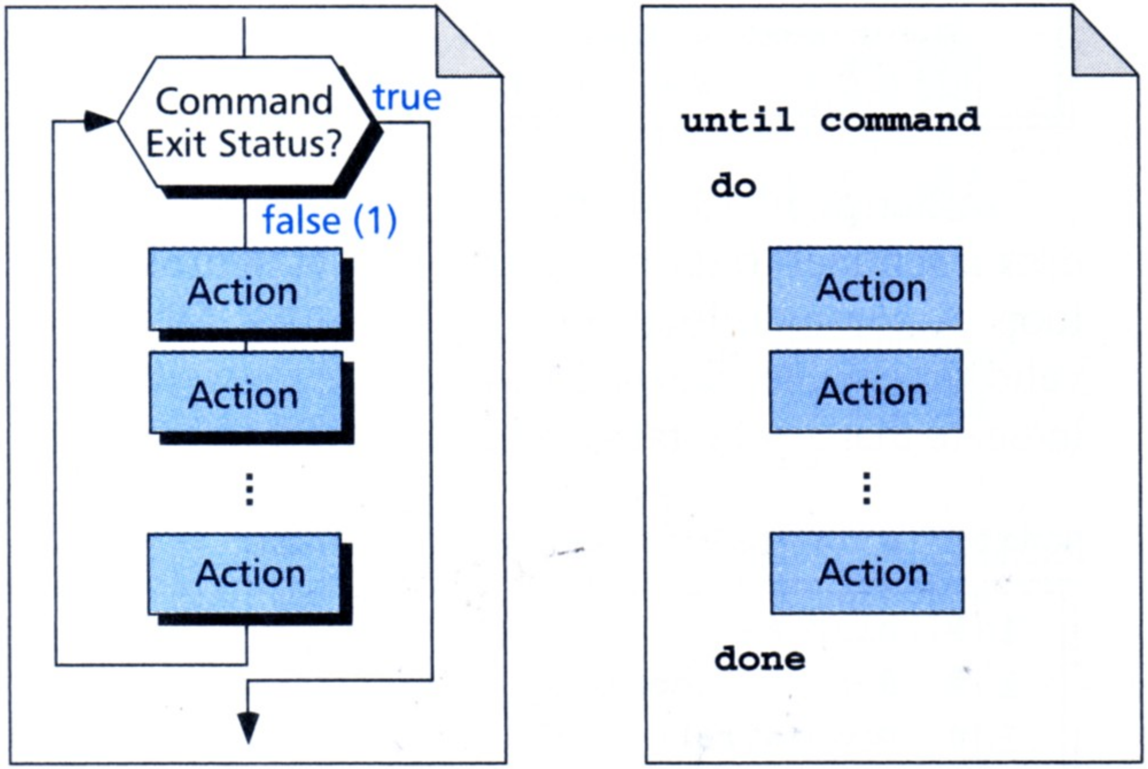
```
while command
  do
      Action
      Action
        ⋮
      Action
  done
```

while: Example I

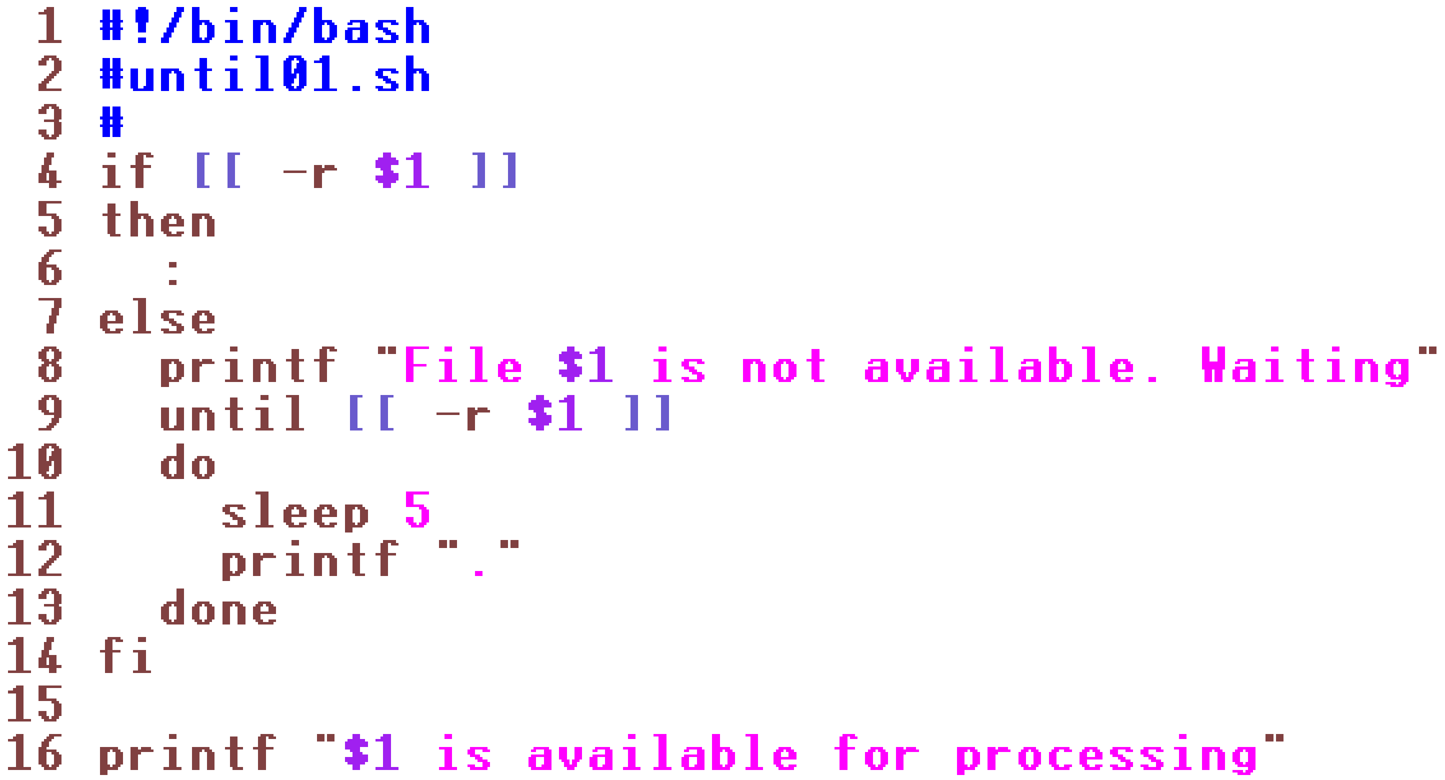


```
#!/bin/bash
#while01.sh
#
echo "This utility adds numbers entered from the"
echo "keyboard. When all numbers have been entered,"
echo "key ^d (eof) to see the total."

sum=0
printf "Enter a number   : "
while read data
do
  (( sum = sum + data ))
  printf "Enter next number: "
done
printf "\n      Sum is: %d" $sum
```

until loop

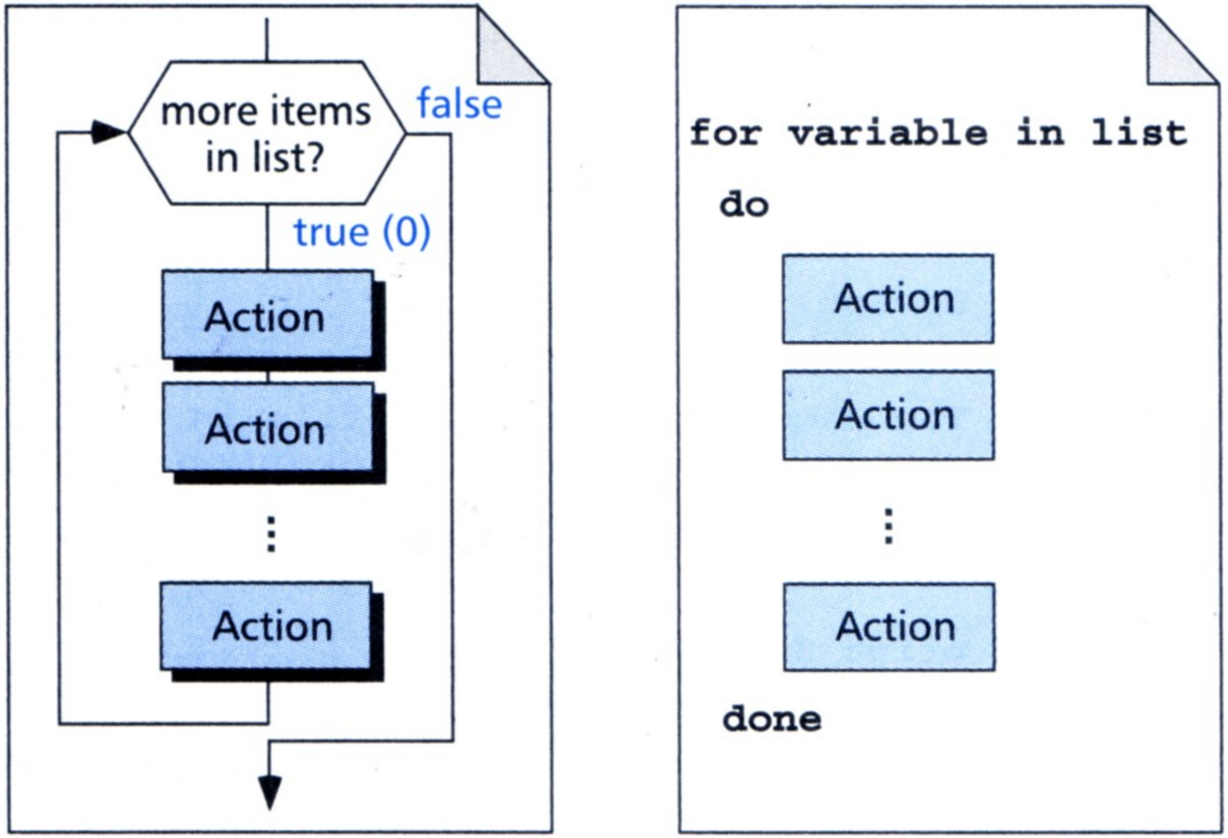

Command
Exit Status?
true
false (1)
Action
Action
⋮
Action
until command
do
Action
Action
⋮
Action
done

until: Example



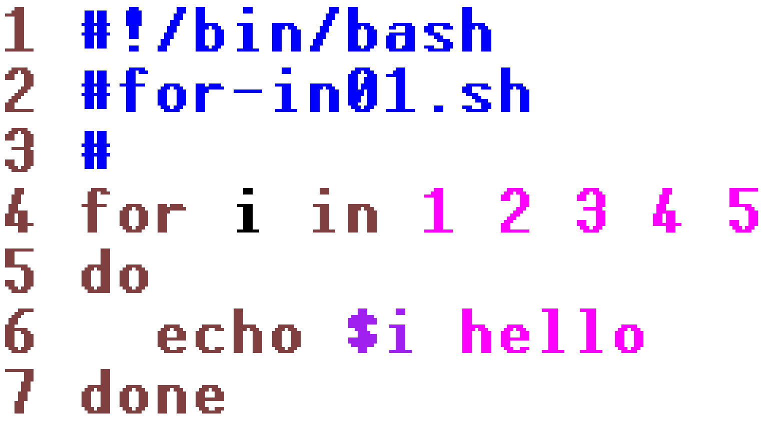
```bash
#!/bin/bash
#until01.sh
#
if [[ -r $1 ]]
then
  :
else
  printf "File $1 is not available. Waiting"
  until [[ -r $1 ]]
  do
    sleep 5
    printf "."
  done
fi

printf "$1 is available for processing"
```

for … in loop

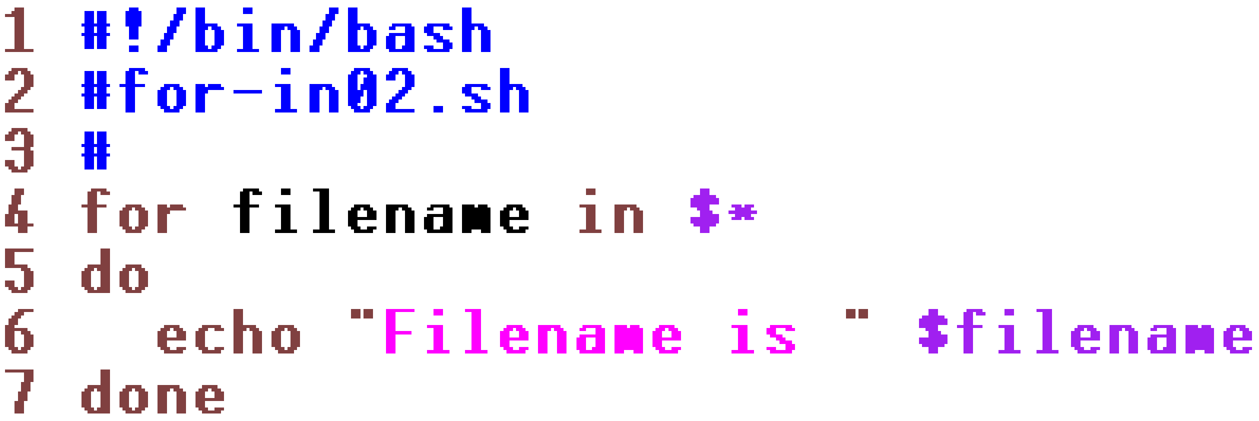


```
for variable in list
  do
      Action
      Action
        ⋮
      Action
  done
```

for ... in: Example

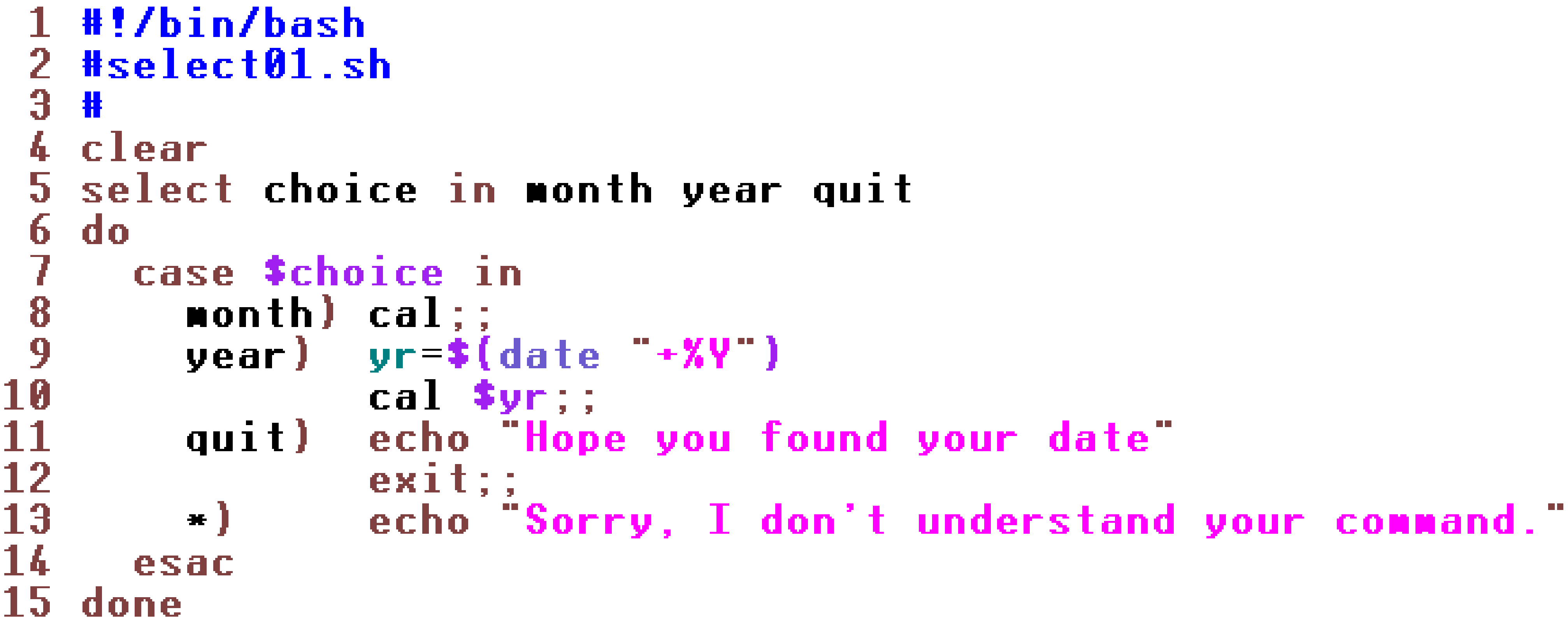


```
#!/bin/bash
#for-in01.sh
#
for i in 1 2 3 4 5
do
  echo $i hello
done
```



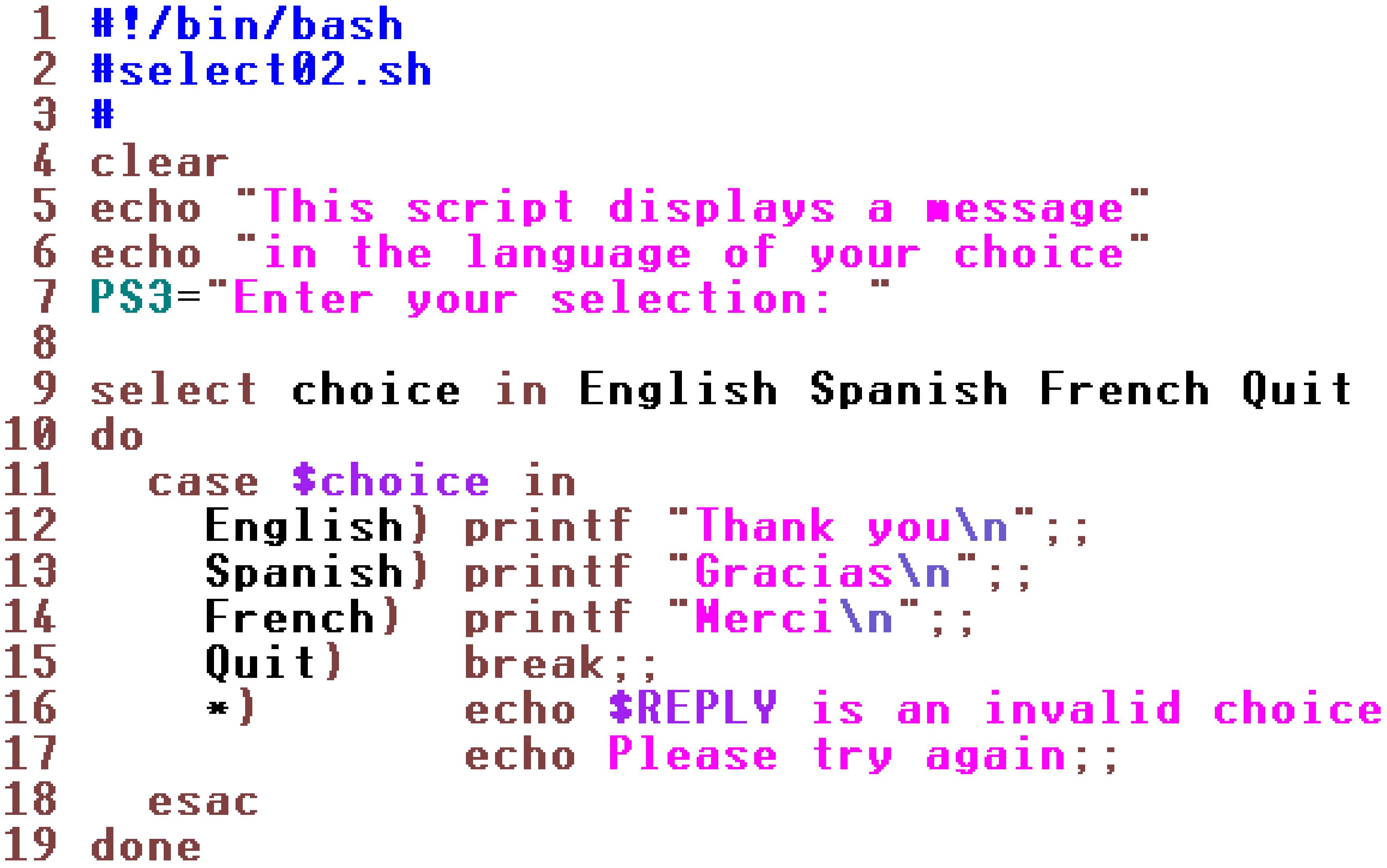
```
#!/bin/bash
#for-in02.sh
#
for filename in $*
do
  echo "Filename is " $filename
done
```

select: Example I



```
#!/bin/bash
#select01.sh
#
clear
select choice in month year quit
do
  case $choice in
    month) cal;;
    year)  yr=$(date "+%Y")
           cal $yr;;
    quit)  echo "Hope you found your date"
           exit;;
    *)     echo "Sorry, I don't understand your command."
  esac
done
```

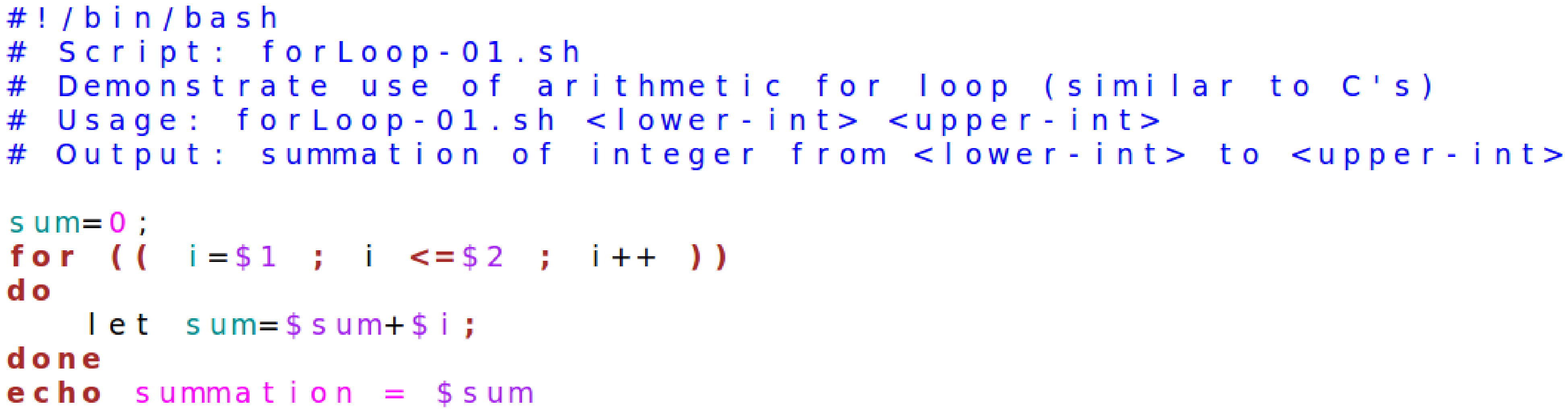
select: Example II



```
#!/bin/bash
#select02.sh
#
clear
echo "This script displays a message"
echo "in the language of your choice"
PS3="Enter your selection: "

select choice in English Spanish French Quit
do
  case $choice in
    English) printf "Thank you\n";;
    Spanish) printf "Gracias\n";;
    French)  printf "Merci\n";;
    Quit)    break;;
    *)       echo $REPLY is an invalid choice
             echo Please try again;;
  esac
done
```

Arithmetic for loops: Example

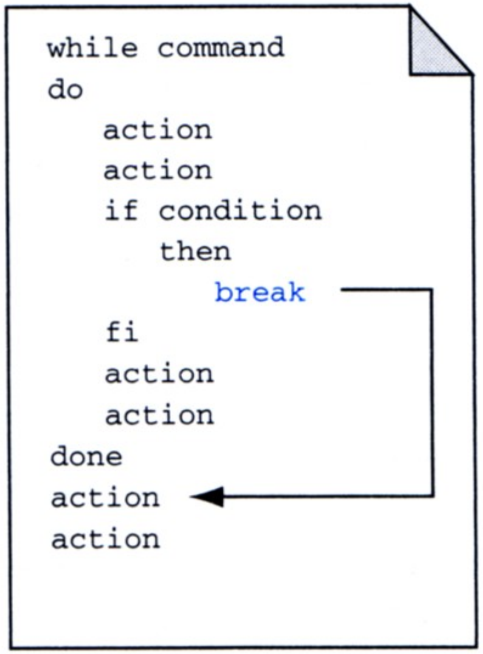
- Useful when dealing with arrays



```
#!/bin/bash
# Script: forLoop-01.sh
# Demonstrate use of arithmetic for loop (similar to C's)
# Usage: forLoop-01.sh <lower-int> <upper-int>
# Output: summation of integer from <lower-int> to <upper-int>

sum=0;
for (( i=$1 ; i <=$2 ; i++ ))
do
    let sum=$sum+$i;
done
echo summation = $sum
```

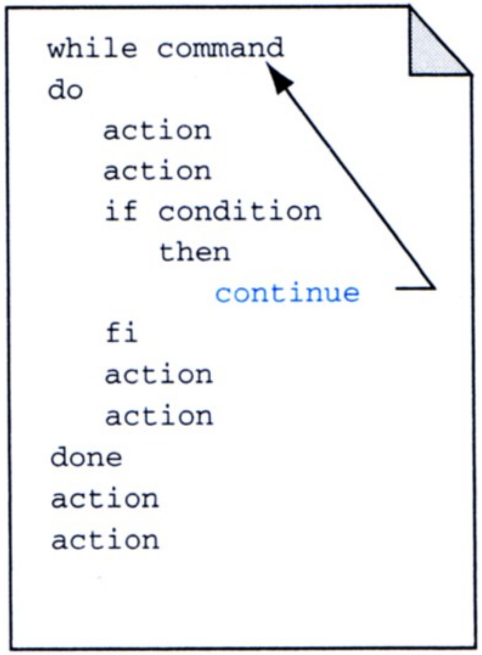
Loop Control Statements

- break and continue



```
while command
do
    action
    action
    if condition
        then
            break
    fi
    action
    action
done
action
action
```

break

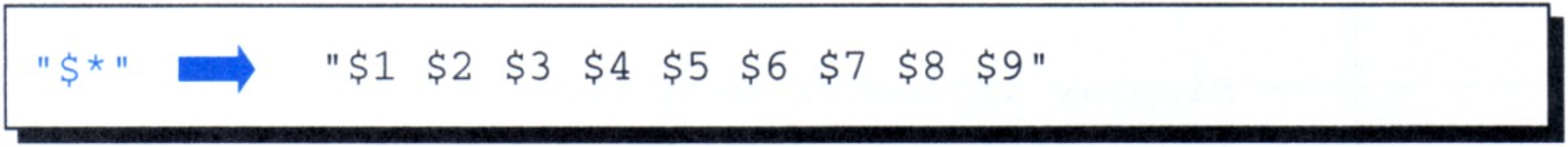


```
while command
do
    action
    action
    if condition
        then
            continue
    fi
    action
    action
done
action
action
```

continue

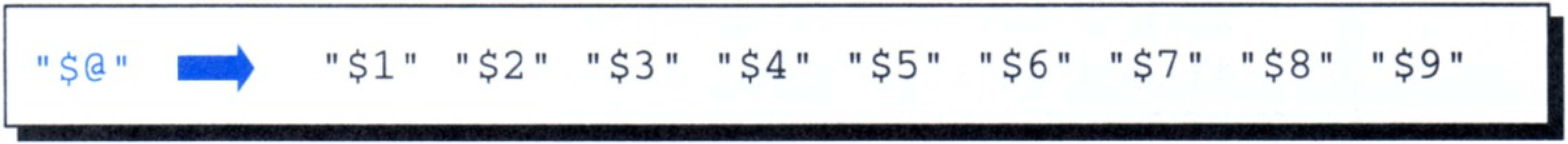
Special Parameters and Variables

- $0 - Script name
- $# - Number of arguments
- $*, $@ - All Parameters



```
"$*"  ➡  "$1 $2 $3 $4 $5 $6 $7 $8 $9"
```

(a) Quoted All Parameters String ("$*")

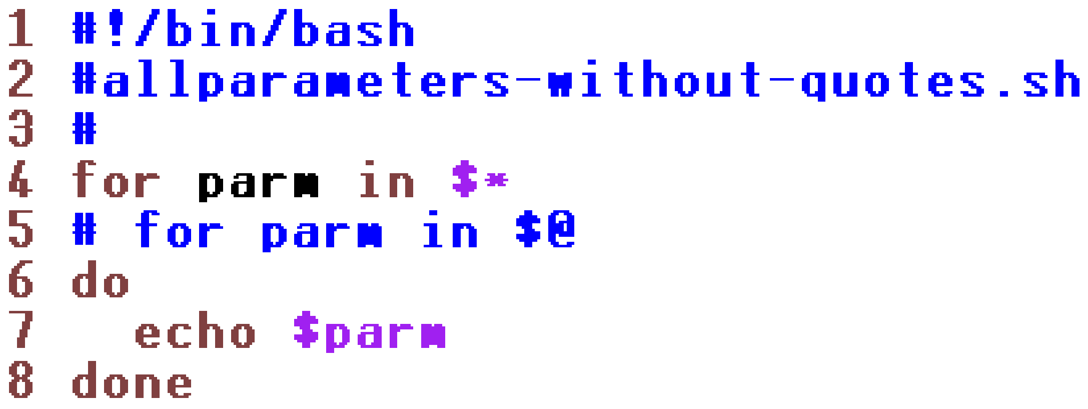


```
"$@"  ➡  "$1" "$2" "$3" "$4" "$5" "$6" "$7" "$8" "$9"
```

(b) Quoted All Parameters List ("$@")

All Parameters: Example I

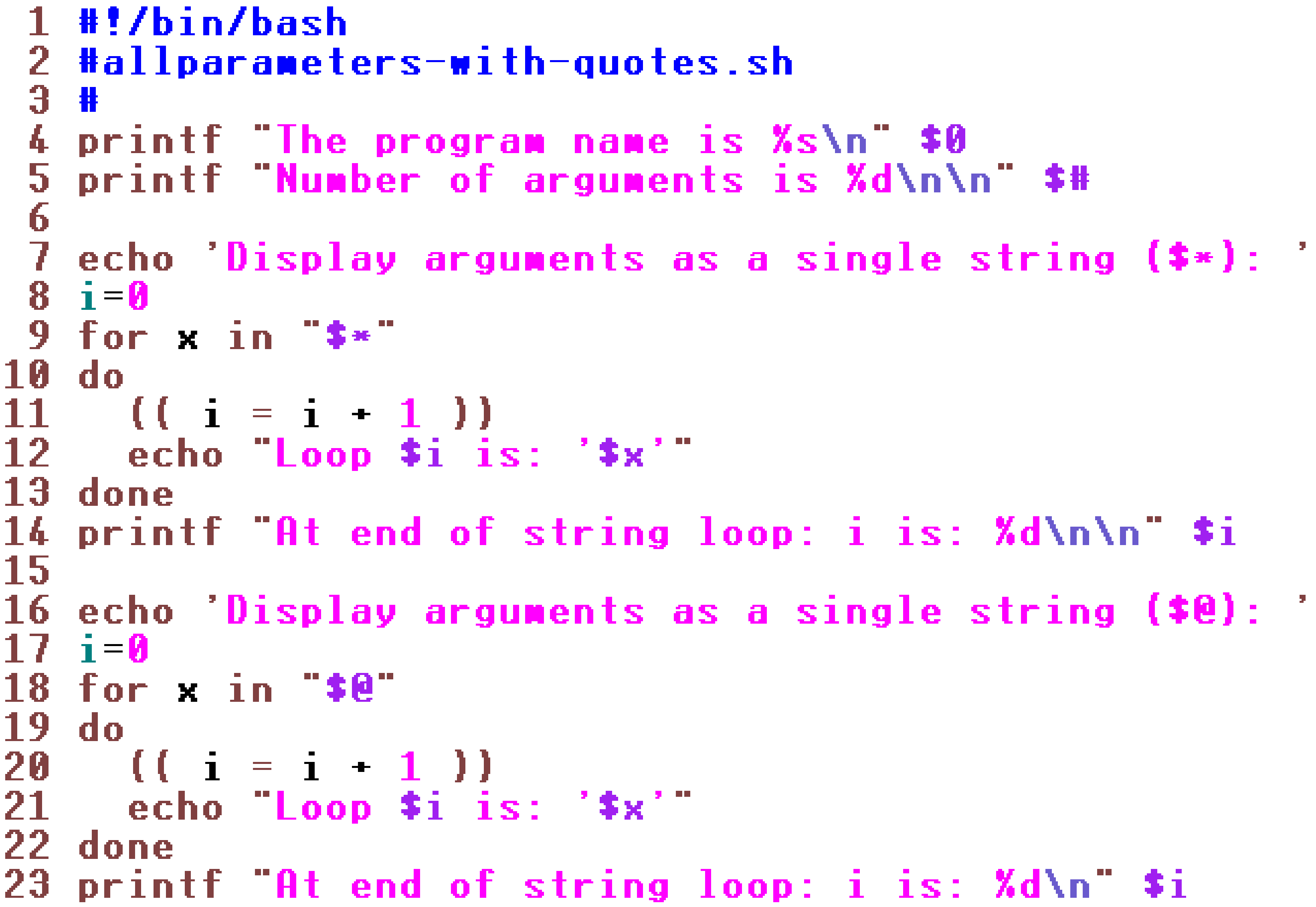
- All parameters without quotes
  - □ $* and $@ give the same result



```
#!/bin/bash
#allparameters-without-quotes.sh
#
for parm in $*
# for parm in $@
do
  echo $parm
done
```

- All parameters with quotes
  - □ “$*” = combine all arguments into one string
  - □ “$@” = create a list of each argument as a separate string

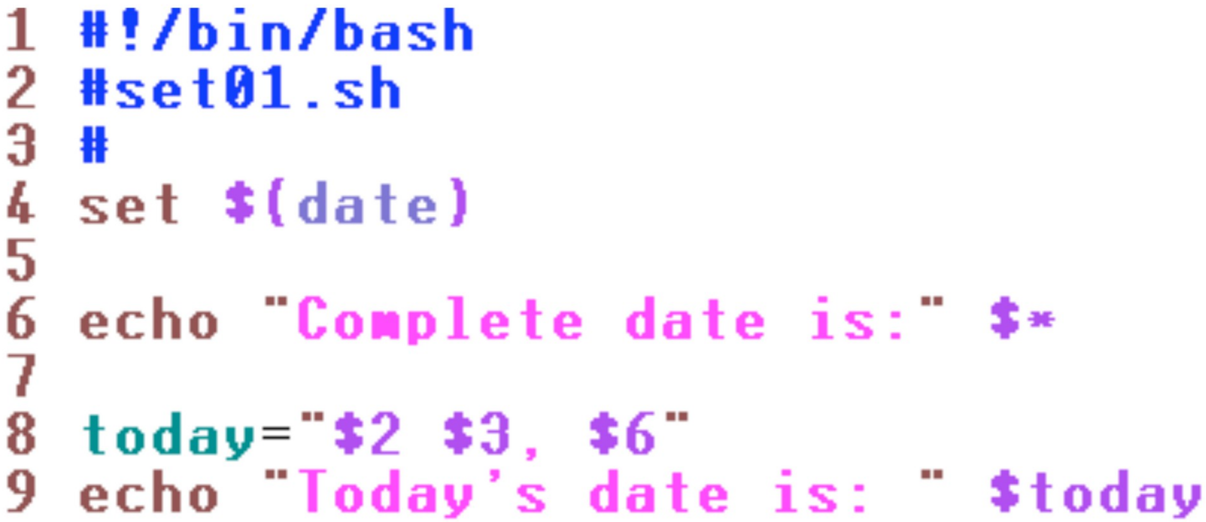
All Parameters: Example II



```
#!/bin/bash
#allparameters-with-quotes.sh
#
printf "The program name is %s\n" $0
printf "Number of arguments is %d\n\n" $#

echo 'Display arguments as a single string ($*): '
i=0
for x in "$*"
do
  (( i = i + 1 ))
  echo "Loop $i is: '$x'"
done
printf "At end of string loop: i is: %d\n\n" $i

echo 'Display arguments as a single string ($@): '
i=0
for x in "$@"
do
  (( i = i + 1 ))
  echo "Loop $i is: '$x'"
done
printf "At end of string loop: i is: %d\n" $i
```

Changing Positional Parameters

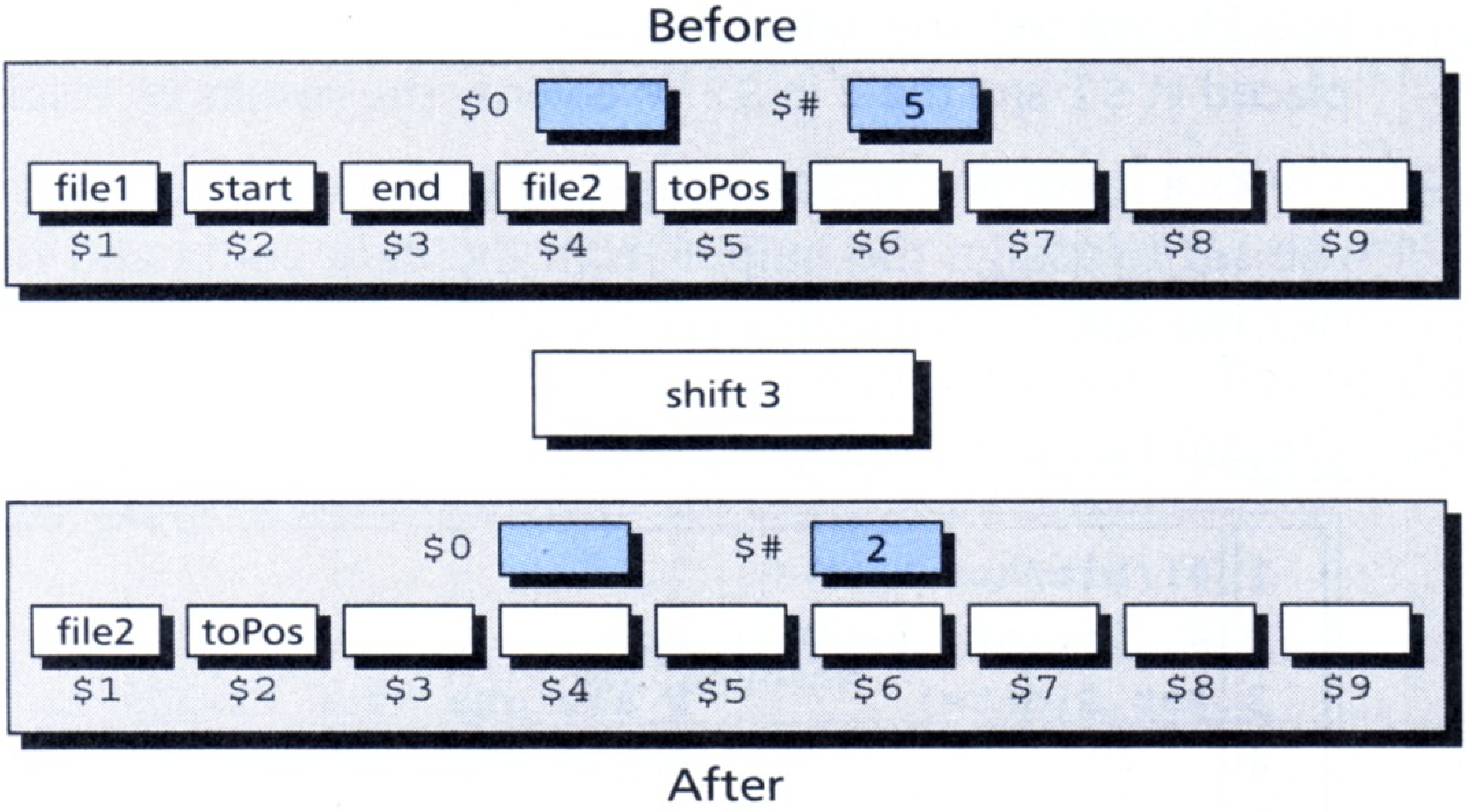
- Positional parameters can be changed with `set` statement



```
#!/bin/bash
#set01.sh
#
set $(date)

echo "Complete date is:" $*

today="$2 $3, $6"
echo "Today's date is: " $today
```

shift Command

- `shift` works by shifting the values of positional parameters to the left, one value at a time



Before

$0 | $# 5

$1	$2	$3	$4	$5	$6	$7	$8	$9
file1	start	end	file2	toPos				

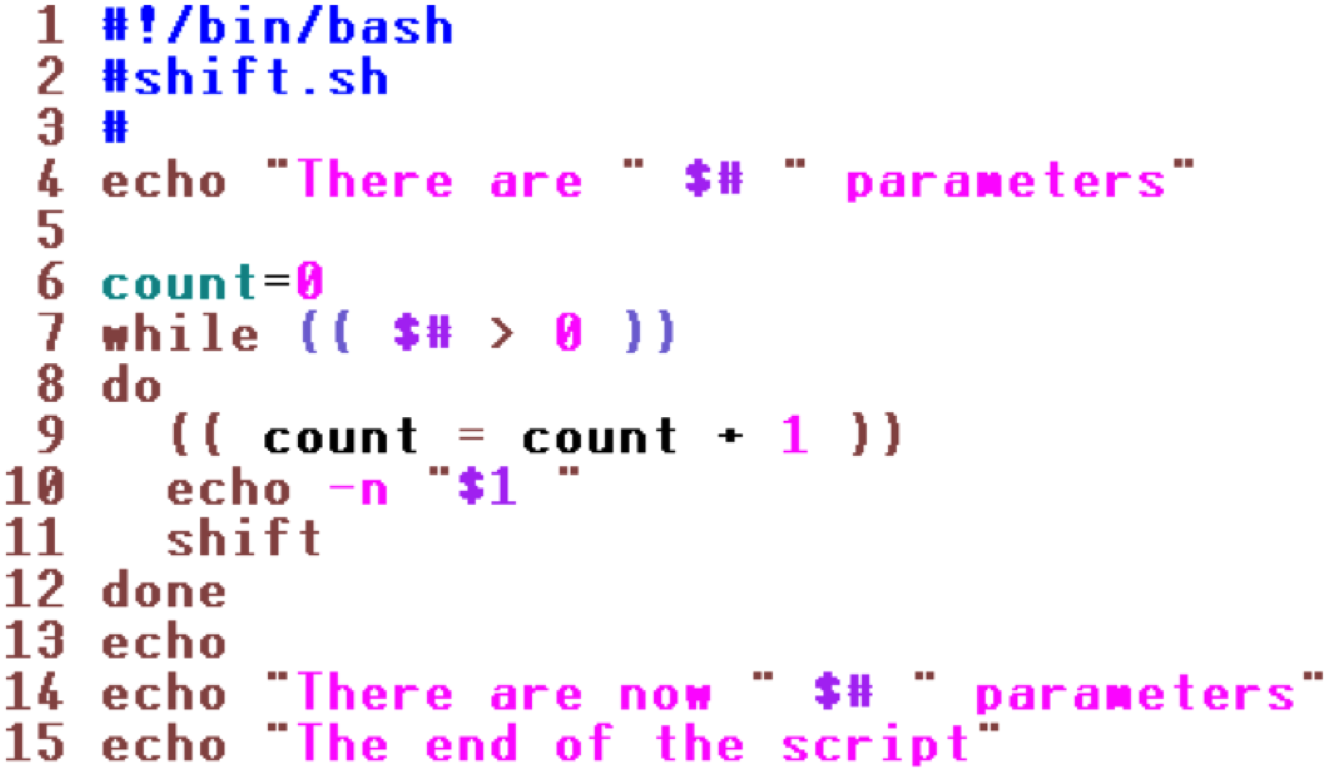
shift 3

$0 | $# 2

$1	$2	$3	$4	$5	$6	$7	$8	$9
file2	toPos							

After

shift: Example



```
#!/bin/bash
#shift.sh
#
echo "There are " $# " parameters"

count=0
while (( $# > 0 ))
do
  (( count = count + 1 ))
  echo -n "$1 "
  shift
done
echo
echo "There are now " $# " parameters"
echo "The end of the script"
```