Command arguments

- typical examples include:
  - `ftp somemachine.co.uk`
  - `cat --help`

- we may want our scripts to handle arguments as well
  - fortunately we can do this

- ```
  #!/bin/sh

  echo $1 $2 $3
  ```

- create a file called `args1.sh` and make this script executable

Command arguments

- if we type: `./args1.sh apple pear orange` then we see the output:

- ```
  apple pear orange
  ```

- so we see that $1 represents the first argument, $2 the second etc

- now we alter script to contain:

- ```
  #!/bin/sh

  shift 1
  echo $1 $2 $3
  ```

Shift

- the shift operator moves all argument values to the preceding argument index

Usage

- every useful script should have a usage description
  - under GNU/Linux the tradition is that this is printed
    - either if the user specified `-h` or `--help`
    - or if the user specified an incorrect argument or no arguments

- let us add a function `usage` to autoftp

Improving autoftp

■
```
#!/bin/sh

scriptname=$0   # $0 is the name of the program

function usage () {
   cat <<EOF
Usage: scriptname [-p] [-v] [-h] filename
   -p   probes the ftp sites and check urls in
        filename
   -v   executes and prints out verbose messages
   -h   displays basic help
EOF
}

if [ "$1" = "-h" ]
then
   usage
   exit 0
fi
```

Better argument handling

■ previous method rather ad hoc
  ■ we might also like to combine arguments so
    ■ user could type: `autoftp -vp autoftprc` rather than
    ■ `autoftp -v -p autoftprc`

■ bash provides us with a better mechanism, `getopts`

getopts

■ takes two arguments
  ■ the first is a string
  ■ second is a variable

■ the string defines the various options and whether they take further arguments

getopts example

■
```
#!/bin/sh

progname=$0

function usage () {
   cat <<EOF
Usage: $progname [-a] [-b name] [-c] etc
EOF
   exit 0
}

while getopts ":ab:ch" opt; do
   case $opt in

   a )  echo "found -a" ;;
   b )  echo "found -b and $OPTARG is after -b" ;;
   c )  echo "found -c" ;;
   h )  echo "found $opt" ; usage ;;
   \?)  usage ;;
   esac
done

shift $(($OPTIND - 1))

echo "the remaining arguments are: $1 $2 $3"
```

getopts example run

- we can run the script `args3.sh` and see the effect of different arguments

- ```
  found h
  Usage: ./args3.sh [-a] [-b name] [-c] etc
  ```

- note that −h was detected and usage called
  - also note that $opt was set to h

- ```
  ./args3.sh -a -c
  found -a
  found -c
  the remaining arguments are:
  ```

getopts example run

- ```
  ./args3.sh -a -c foo bar mumble
  found -a
  found -c
  the remaining arguments are: foo bar mumble
  ```

- ```
  ./args3.sh -a -b file foo bar mumble
  found -a
  found -b and file is the argument to -b
  the remaining arguments are: foo bar mumble
  ```

- note that the line `shift $(($OPTIND - 1))` removes all arguments that have been processed by `getopts`

the getopts string

- `":ab:ch"`

- initial colon
  - means getopts should not produce an internal error message if the user passes an option which is not declared in the string

- `a` means −a is legal

- `b:` means −b is legal
  - furthermore −b takes an additional parameter
    - obtained via: $OPTARG

- `c` means −c is legal
  - h means −h is legal

getopts justification

- `getopts` may seen complex, but we can just use the above template and modify it slightly whenever we need to manipulate command arguments

- the getopts mechanism appears in other languages, C
  - learn once use many times..

Tutorial exercise

- improve your copy of autoftp to include command line argument handling using getopts
  - it should set probeonly=yes if the −p option was specified
  - it should set verbose=yes if the −v option was specified
  - −h should be honoured

- remember to initialise the variables at the beginning

autoftp with getopts

-
```
#!/bin/sh

scriptname=$0    # $0 is the name of the program
verbose=no
probeonly=no

function usage () {
   cat <<EOF
Usage: $scriptname [-p] [-v] [-h] filename
   -p   probes the ftp sites and check urls in
        filename
   -v   executes and prints out verbose messages
   -h   displays basic help
EOF
   exit 0
}

while getopts ":pvh" opt; do
   case $opt in

   v )   verbose=yes ;;
   p )   probeonly=yes ;;
   h )   usage ;;
   \?)   usage ;;
   esac
done

shift $(($OPTIND - 1))

echo "verbose=$verbose probeonly=$probeonly"
echo "remaining arguments=$*"
# etc etc
```

example execution

-
```
./autoftp3.sh -v
verbose=yes probeonly=no other args=
```

-
```
./autoftp3.sh -v -p foo bar
verbose=yes probeonly=yes other args=foo bar
```

-
```
./autoftp3.sh -h
Usage: ./autoftp3.sh [-p] [-v] [-h] filename
   -p   probes the ftp sites and check urls in
        filename
   -v   executes and prints out verbose messages
   -h   displays basic help
```

Laboratory work

- try out the examples given in this section of the notes and extend the examples to include more arguments

- add the usage command to your assignment