Command line arguments

- Bash scripts can accept command line arguments, i.e. you can pass parameters to a script when you run it (e.g. ./myscript.sh hello 1 2 3 “and this” 27)
- Arguments are stored in an array named $@, and (as with function parameters) the variables $1, $2 etc access the parameters by position, $0 gives the script name, $# gives the number of parameters

Iterating through the arguments

- We can iterate through the command line arguments using loops (again, as we did with function parameters), e.g.

```
for arg in “$@”; do
    echo “the next argument is $arg”
done
```

Shift and iteration

- “shift N” is also built in, and effectively removes the “front” N elements from $@ (doesn’t change $0, and the default value for N is 1)
- Here we’ll print and shift out one element at a time

```
while [ $# -gt 0 ] ; do
    echo “$1”
    shift 1  # or just shift, since it defaults to 1
done
```

Passing the arguments

- When we run the script, each “word” on the command line is treated as one argument, e.g. ./myscript 1 2 foo blah
- We can group multiple words together as one argument with quotes. e.g. ./myscript.sh “arg one” argtwo
- Use single quotes if you’re passing special characters that you don’t want bash to interpret before they reach the script, e.g. ./myscript.sh ‘foo*’