Shell command line substitutions (expansions) revisited

...‘*command*‘... — substitute the output of a *command*
*, ?, [ ] — substitute all matching filenames
{} — substitute all choices
\ — escape next char
$*var* — substitute value of variable *var* (try `echo $PATH`)
"..." — only variable and command expansion
'...' — pass everything as-is

Combining commands (revisited)

command1; *command2* — run commands sequentially
command1 && *command2* — run *command2* if *command1* succeeded
command1 || *command2* — run *command2* if *command1* failed
(*command1*; *command2*; *command3* ...) — run commands in a *subshell*
: — do nothing

Shell scripts

When a first line in a text file has a form
`#!`*/full/path/to/program options*
and the file is executable, than running *file arguments* is the same as running
/full/path/to/program options file argument

Such a file would be called a script.

Shell scripts: $0 is a script name, $*i* is an *i*-th argument ($1 <= i <= 9$), $* is all arguments

Bourne shell scripts

Branching

```
if condition1
then command1
elif condition2
then command2
...
else command3
fi
```

`for` loop

```
for var in list
do
   commands
done
```

`while` loop

```
while condition
do
   commands
done
```

`if` *condition*; `then` *command*; `fi` is the same as
condition && *command*

`if` *condition*; `then :; else` *command*; `fi` is the same as
condition || *command*

Conditions — `test`

`test` *condition* or `[` *condition* `]`

Condition	Meaning
`-f` *file*	*file* exists and is a regular file
`-r` *file*	*file* exists and is readable
`-d` *file*	*file* exists and a directory
`-n` *string*	*string* has non-zero length
`-z` *string*	opposite of `-n`
s1 `=` *s2*	strings are identical
n1 `-gt` *n2*	*n1* is greater than *n2*
n1 `-lt` *n2*	*n1* is less than *n2*
`(`*condition*`)`	same as *condition*
condition1 `-a` *condition2*	both are true
condition1 `-o` *condition2*	either one is true