If statements

- if statements can use the [ ] for simple expressions, e.g.

```
x=3
if [ $x -gt 2 ] ; then
    echo “$x is bigger”
elif [ $x -lt 2 ] ; then
    echo “$x is smaller”
else
    echo “$x is 2”
fi
```

Comparison operators

- The operators == != < > etc are used for STRING comparison, not integers
- The operators -ge (greater than or equal), -gt, -lt, -le, -eq, and -ne are used for number (integer) comparisons
- Note bash can be pretty picky with the whitespace syntax around the [ ]

Compound expressions

- The ! acts as the not operator, e.g.

```
if ! [ $x == $y ] ; then
```

- For compound expressions in single square brackets, -a is used for and, -o is used for or, e.g.

```
if [ $x -eq $y -a $x -gt $z ] ; then
```

Unary expressions

- A variety of special operators are supported by test, largely focused on testing file/dir properties, here assuming $fname is supposed to hold the name of a file or directory:
  - [-f $fname] is true if the file exists
  - [-d $fname] is true if directory exists
  - [-r $fname] is true if it is readable
  - [-w $fname] is true if it is writeable
  - [-x $fname] is true if it is executable
  - [-n $fname] is true if the filename isn’t null

Double square brackets

- The || and && syntax for compound boolean expressions is supported by the [[ ]] syntax, e.g.

```
If [[ $x == $y && $z == $w ]] ; then
```

- The double-bracket syntax is way more flexible, I would tend to recommend that be your syntax of choice
- The [[ ]] syntax is also used for regular expression matching, but we'll come back to that later