THE UNIVERSITY OF WESTERN AUSTRALIA

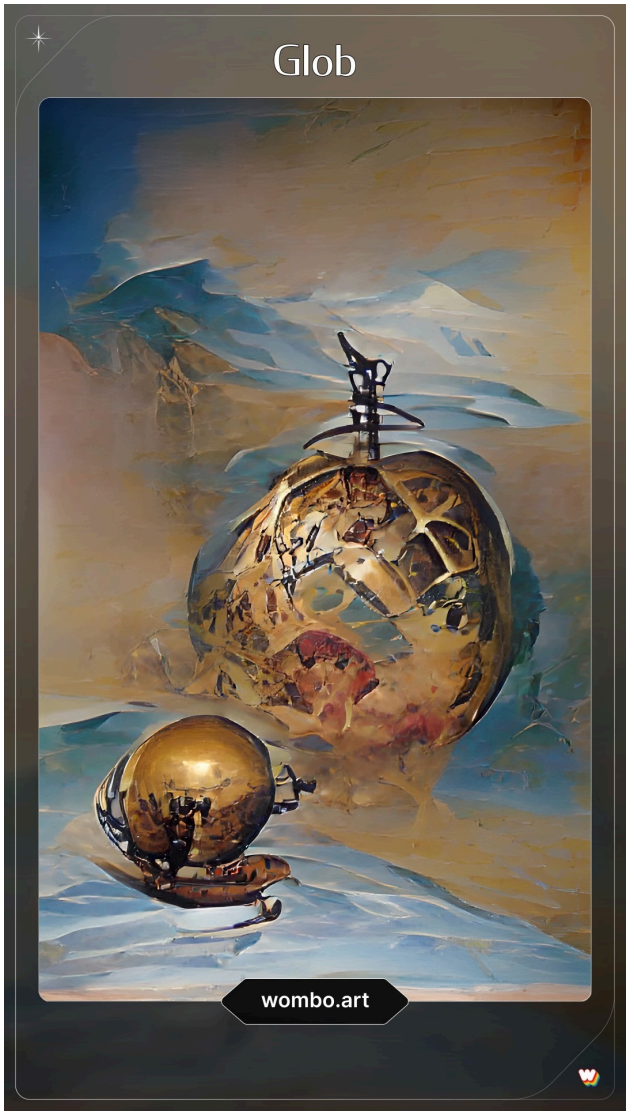
Globbing, case, for

Lecture 7

Michael J. Wise

Globbing/wildcards

- Globbing is just another way of saying match-anything (aka wild-card).
- ***** matches *anything* in a filename, e.g. `ls x*` matches: `x`, `x.bak`, `x.txt`, `x1`, `xfer_peter` (The last is a directory)
- **?** Matches a single letter. In the list above `x?` only matches `x1`



Globbing/wildcards

- `[` *<letters>* `]`
  - *Match any **one** of these letters (can be a range)*
- `[!` *<letters>* `]`
  - *Match any **one** letter BUT NOT any of these (can be a range*

```
% ls [a-z].*
```

a.awk a.sed x.bak x.txt x.xml

```
% ls [!x].*
```

returns, what?

case

- Case implements a multiple choice test, looking for a pattern match.

`case` <expression> `in`

<*pattern*> [**|** <*pattern*> ....] **)** <*statements*> **;;**

<*pattern*> [| <pattern> ....]) <statements> ;;

.....

not necessary

`esac`

- The <*expression*> is anything (i.e. variable or command) that returns a string
- Each <*pattern*> can be of the sort used in file-name generation, including wild-cards.
- Alternate patterns, but with the same set of actions, are separated by |.

case

- The expression is evaluated and the result tested against the patterns top-down, left-to-right across alternative patterns separated by `|`.
- If a match, the corresponding statements are executed (up to the `;;`).
- If no patterns match, execution proceeds to the next statement
- `*` matches every string, so is used as the default pattern (i.e. just like `else`).

case

```
case $DAY in
   Mon | Fri ) echo ${DAY}day ;;
   Tue | Thu ) echo ${DAY}sday ;;
   Wed ) echo ${DAY}nesday ;;
   S??) echo WEEKEND! ;;
   *) echo "$DAY is not a day I understand."
      echo "April Fool's Day?" ;;
esac
```

for

`for` <*name*> `in` <*list*>

`do`

    <*commands*>

`done`

- <*list*> is just a white-space list of strings
- At each iteration, the variable <*name*> is assigned the next item in the list.

```
for i in *.[ch]      # * at shell level is list of files
do                   □list□files□with□names□consisting□of□a□single□character□followed□by□either□'.c'□or□'.h'.
   echo $i
   diff $i ../tempdir/$i
   echo
done
```

for can be used on files

The for loop can also be used to read through text files

```
for word in $(< file)
do
……
done
```

Returns each word. To return lines need to change Shell variable IFS (Internal Field Separator)

```
IFS="     # This captures a new line
"
for word in $(< file)
```

…. *etc* …

Demo

- What does this code do?

```
for i in one 2 "2 1/2"
do
   echo "$i:${#i}"
done
```

Demo

- What does this code do?

```
for i in *
do
   echo "$i:${#i}"
done
```

Demo

```
IFS="
"
for i in $(< Alice_in_Wonderland.txt)
do
..... <etc>
```

Can be used to go through the lines in a text file, one at a time (`i` is set to first line, then the second line, etc).

- Write a script, `longest_line` which, given a text file, reports the longest line and its length.