Lecture 3:

Sep 20, 2016

CS102, Fall 2016

Overview

A. Hw2

B. Makefile and Shell

Next...

Hw2

Makefiles + Shell

[Start] [End]

Hw2

Cash ATM

Remarks about ATM solutions (Fred)

Go over CashAbstract.java overview

Cash ATM

Hw2 Demo on Eclipse

Hw2 Demo on Terminal

- How to redirect input ($<$)
- How to redirect output ($>$)

- Bit of theory: processes has three standard I/O files:

  stdin, stdout, stderr

For more info: https://en.wikipedia.org/wiki/Standard_streams

Cash ATM

CHANGES needed in your CashAbstract.java

- replace new CashAbstract() by new Cash()
- replace split(" ") by split("[\t]")

More on Shell

Q: Who is interpreting your commands in a Terminal?

More on Shell

A: The “shell” program!

Q: What happens when you type “cd”?

More on Shell

A: Shell looks in folders in PATH for “cd”

– try “which cd” to see where it is found

Q: How to call Chrome browser from the Terminal?

More on Shell

A: 2 steps:

- 1. Find its location in C:
- 2. then link from a folder of PATH

HINT: link /cygdrive/c to /c and link /cygdrive/c/Users/yap to /yap

More on Shell

Shells have “states” information

- Use export and printenv to set and get these values.
- You can set these in shell initialization (e.g., in .profile)

More on Shell

Shell scripts (plain text files)

First line in script is #!/bin/bash

DEMO – ssh to my Courant Account

E.g., scp2 – copy files to my Courant Account (say ”zzz”)

E.g., scp4 – copy files from my Courant Account (say ”xxx”)

Next...

Hw2

Makefiles + Shell

[Start] [End]

II

Makefiles + Shell

Targets and Actions

Study sample Makefile:

* A Makefile is a plain text file
* Comment character is #
  - the comment character and rest of line is ignored

Targets and Actions

Basic terms:

targets, actions

Target lines has (at least) one target name followed by a colon ':'

– Following the target line are zero or more action

– After colon, there is an optional list of dependencies

– Names on the same target line are alternatives lines.

Targets and Actions

How to call a target?

* Assume the Makefile in the current directory has a target called compile
* You type on your Terminal the command

  $\gg$ make compile
* All the actions of compile will be executed!

Dependencies

They can be target names or file names!

E.g., you can have a target like this:

ATM.class: ATM.java

Need a target called ATM.java!

Variables

Assignment statements

variableName = value

E.g.,

n=123

args="1 two 3"

Delayed recursive evaluation:

args=$(_args)

_args="1 two 3"

@echo $(args) — what does it show?

* SO, do not write: args=$(arg1) $(args)

Variables

Overriding the variable assignment at the terminal

- E.g., $\gg$ make run n=456
- One of the main tricks we exploit to do testing

Dependencies, conditionals

They can be target names or file names!

– they are placed right after the colon

– non-file targets are called pseudo targets

Dependencies, conditionals

How does make evaluate the Makefile?

– in two phases:

1. It first sets up the Make variables
2. Then execute the actions by calling shell

Dependencies, conditionals

E.g., you cannot do this (this is phase 1):

```
ifndef MYFLAG
     echo "not defined!"
else
     echo "defined!"
endif
```

Solution?

Dependencies, conditionals

```
ifndef MYFLAG
     _MYFLAG="not defined!"
else
     _MYFLAG="defined!"
endif

zzz:
     @echo $(_MYFLAG)
```

Thanks for Listening!

“Algebra is generous,
she often gives more than is asked of her.”

— JEAN LE ROND D’ALEMBERT (1717-83)