

Introduction to Makefiles

CPSC 2150

Creating a Java Program (on Unix)

- Create a file just like you would with C++ on Unix but use the `.java` file extension
 - **Ex.** `HelloWorld.java`
 - It is common to see Java programs named with uppercase letters
- To compile the program, use the `javac` command
 - **Ex.** `javac HelloWorld.java`
 - Like there is with the C++ compilers, there are many flags and options for the Java compiler. We'll cover those as we need to
- To run your program, use the `java` command
 - **Ex.** `java HelloWorld`
 - **Note:** now there is no file extension

Compiling Complications

- When using packages, our commands get a little longer
 - We have a folder for the project, called `myProject`
 - Package is `cpssc2150.lab1`
- Code is in the `myProject/cpssc2150/lab1/` directory and we are currently in the `myProject` directory
- Compile:
 - `javac cpssc2150/lab1/HelloWorld.java`
- Run: (*Note:* the command uses dots rather than slashes)
 - `java cpssc2150.lab1.HelloWorld`
- Longer, but still not too bad

Compiling Complications

- What if we have multiple files in our project
- Same file structure as before
- Compile:
 - `javac cpssc2150/lab1/Menu.java cpssc2150/lab1/Order.java cpssc2150/lab1/Pizza.java cpssc2150/lab1/Topping.java cpssc2150/lab1/Customer.java`
- Run:
 - `java cpssc2150.lab1.Menu`
- This is starting to be a pain
- We can use the up arrows in Unix, but we still have to retype every once in a while

A solution

- We're computer scientists, shouldn't we have a way to automate this?
- Enter **makefiles**
- Create a file just named "makefile"
 - No extension
- Put your compile and run statements in `makefile`
 - With some specific syntax
- Put `makefile` in your `myProject` directory
- Type "make" to compile
- Type "make run" to run the program
- Type "make clean" to delete the compiled files
 - This is avoid turning in the compiled files, rather than your source code (`.java`) files

Makefile Syntax

<target>: **<dependencies>**

<tab>**<command>**

<target> - can name it what you want, but there are some standards: `default`, `run`, `clean`

<dependencies> - what files are needed? Can be other targets with their own compilation statements

<tab> - Seriously, that tab needs to be there.

<command> - your compile statement

Example: (*Note:* `default` goes on the top of the file)

```
default: cpsc2150/lab1/HelloWorld.java
    javac cpsc2150/lab1/HelloWorld.java
```

Makefile `run` Syntax

- Same syntax as before, but what we plug in changes
- `<target>` - name it `run`
- `<dependencies>` - the name of your compiled file
- `<command>` - your run command
- **Example:**

```
run: cpsc2150/lab1/HelloWorld.class  
    java cpsc2150.lab1.HelloWorld
```

Makefile `clean` Syntax

- Helpful to get rid of the `.class` files, so you don't accidentally submit those
- Target is "`clean`"
- No dependencies
- Command is Unix's command to delete files with `.class` extension. (Note: The following example assumes the `.class` files are in the same folder as the `makefile`. If you have packages, then you will need to provide the path to the `.class` files)

`clean:`

```
rm -f *.class
```

Using Your Makefile

- Typing “make” will run the top target (which is why it should be default)
- Typing “make <target>” will run another target
 - `make run` – runs the “run” target that runs your program
 - `make clean` – runs the “clean” target that will delete your extra files
- **Note 1:** If your make file gives you an error, type the command directly in the terminal to determine if there any issues with it. Sometimes we think the `makefile` target is not working, when it is really a compiler error.
- **Note 2:** If your command works directly on the terminal, but not when you use your `makefile` target, check to see if you have the following errors:
 - Used spaces rather than the required tab
 - Check you have the correct dependencies listed.
 - Have compiled all the required files.

Advanced Makefile Syntax

- There are a lot of advanced tricks for **makefiles**
- They are helpful for more advanced programs
- We may cover later in the semester
 - If not, there are a ton of good resources online.