Data Structures, Python I/O and Makefiles

Week 3

Craig Rasmussen (Research Support Services, University of Oregon)

Which one did you want? Which is faster?

A *makefile* maintains groups of programs based on dependencies being satisfied

---

```
#
# this is a comment

#
# define environment variables (compilers/linker/libraries...)

CC = gcc

#
# define targets

all: hello

hello.o: hello.c
	$(CC) -c hello.c -o hello.o

hello: hello.o
	$(CC) -o hello hello.o

# run tests
check:

# clean up
clean:
	rm -f hello.o hello
```

target

dependency

tab

Shell Command: *nm*

- nm - display name list (symbol table)
- Steps to build an executable from a source file (*.f90)
  - compile program -> $(FC) -c -I include_path hello.f90
  - link program -> $(FC) -o hello hello.o -L library_path -lsome_library
- hello.o is an object file and contains symbols (e.g., functions) and code
- hello is an executable file (created by linker from *.o and libraries)
- What happens if a symbol (function code) can’t be found by linker?
  - linker can’t create an executable if all dependencies aren’t satisfied
  - use nm to track down missing symbols

Debugging Programs

- Print statements handy
  - especially when you are debugging a parallel program
- But debuggers are great
  - compile with -g option
  - run with gdb (or other debugger, lldb on mac)
  - set break points
  - example variables
  - step through program

Classes

- A class encapsulates functions and state variables
- Class Foo
  - int x, y, z; // state variables
  - void f1(); // function
- A class is a template (recipe) for creating objects
- A program can have pointers to many live objects at once
- Each object contains state
- In parallel programming state is evil!
  - who modified x and when?

Functions

- A function takes input and produces output
- Functions are composable
  - *f3(f2(f1(x)))*
- What happens to the state variables?
  - *f2()* consumes the output of *f1()*
- Going stateless is good!

Unix pipes

- A unix shell program takes input and produces output
    - standard input (file)
    - standard output (file)
- Unix shell programs are composable with pipes
    - program1 | program2 | program3
    - the output of program1 is said to be “piped” to the input of program2
- What happens to the state variables (files)?
    - program2 consumes the output of program1

Computational Complexity Theory

- The complexity of an algorithm is how the runtime scales as the number of elements *N* in a collection (an array for example) increases
- $T(N) = a_0 + a_1 x N^1 + a_2 x N^2 + \ldots$
  - the complexity is the superscript of the leading term
  - call big O notation
- Array access is constant, $O(N^0)$
- The inner product to two vectors is $O(N^1)$
- Building a correlation matrix is $O(N^2)$
- The order of an algorithm using an array data structure is *the number of loops passing over the entire array*