CS143A
Principles of Operating Systems
Discussion 08: Project 1 Part 2

Instructor: Prof. Ardalan Amiri Sani
TA: Ping-Xiang Chen (Shawn)

Agenda

- Makefile
- Tips to navigate a huge codebase
- Useful resources

Makefile

- A Makefile is essentially a text file that automates the process of compiling and building software.
- Dependencies
  - It defines dependencies between files, so the make utility knows which files need to be recompiled when changes occur
- Rules:
  - A Makefile contains a set of rules that specify how to build target files from their dependencies

A Makefile consists of a set of *rules*. A rule generally looks like this:

```
targets: prerequisites
  command
  command
  command
```

Take Makefile of Pintos for example

- Automatic Variables
  - $@
    - The file name of the target of the rule
  - $<
    - The name of the first prerequisite.
- For more information, please check:
  - https://www.gnu.org/software/make/manual/html_node/Automatic-Variables.html

```
# -*- makefile -*-

all:

include Make.vars

DIRS = $(sort $(addprefix build/,$(KERNEL_SUBDIRS) $(TEST_SUBDIRS) lib/user))

all grade check: $(DIRS) build/Makefile
	cd build && $(MAKE) $@
$(DIRS):
	mkdir -p $@
build/Makefile: ../Makefile.build
	cp $< $@

build/%: $(DIRS) build/Makefile
	cd build && $(MAKE) $*

image: build/kernel.img

qemu: build/qemu

qemu-nox: build/qemu-nox

clean:
	rm -rf build
```

Tips to navigate a huge codebase

- ctags
  - A tool that generates an index (or "tags") file of names found in source and header files of various programming languages.
  - This index allows programmers to quickly and easily locate definitions of functions, variables, classes, and other code elements

ctags

- Generate tags file
  - $ cd ~/Pintos/pintos/src
  - $ make tags
- Add the text on the right to your ~/.vimrc
  - $ vim ~/.vimrc
- You should be able to navigate the code by using ctrl +]
- To jump back to previous tags, use ctrl + t

```
"--------------"
"let vim search the tag file from ctags
"--------------"
set tags=./tags,tags;/
```

grep

- grep
  - grep searches for PATTERNS in each FILE
  - $ grep -nr “boot” ./
    - search the word “boot” recursively in current working directory with printing line number with output lines.

```
GREP(1)                                  User Commands                                  GREP(1)

NAME
       grep, egrep, fgrep, rgrep - print lines that match patterns

SYNOPSIS
       grep [OPTION...] PATTERNS [FILE...]
       grep [OPTION...] -e PATTERNS ... [FILE...]
       grep [OPTION...] -f PATTERN_FILE ... [FILE...]

DESCRIPTION
       grep  searches  for PATTERNS in each FILE.  PATTERNS is one or more patterns separated by newline characters, and grep prints each line
       that matches a pattern.  Typically PATTERNS should be quoted when grep is used in a shell command.

       A FILE of “-” stands for standard input.  If no FILE is given, recursive searches  examine  the  working  directory,  and  nonrecursive
       searches read standard input.

       In  addition,  the variant programs egrep, fgrep and rgrep are the same as grep -E, grep -F, and grep -r, respectively.  These variants
       are deprecated, but are provided for backward compatibility.
```

Useful resources

- Vim tutorial
  - Become a vim master
- How does the program work? IA32 programming
  - Computer Systems: A Programmer's Perspective
    - You can find the pdf online (Recommend to read Chapter 3: Machine-Level Representation of Programs)
- Detailed documentation of Pintos
  - Pintos by Ben Pfaff
- From assembly to kernel
  - x86 Assembly: Hello World!
  - Make an OS (x86)
- Previous discussion
  - https://ics.uci.edu/~ardalan/courses/os/discussions/os_discussion_4.pdf

Thank you. Any Questions?