Nicholas LaJoie
ECE 331
2/16/17

Sheaff

- Regex tool:

  qr ⇒ ex.) qr/\d{4}-\d{2}-\d{2}/

  makes it easy!

  "quote reg exp", can assign it to variable

[Exam 1: Two weeks from today!]

- Next HW:
  - C float
  - perl scripting, c code (gonna need for testing for kernal driver)

- hashes in perl
  - associate
  - { } it's a hash (assessing)
  - hash of scalars, not a hash of lists
    - ⇒ Prob solved w/ "references", use \ to define variable as a reference
    - ⇒ dereference ⇒ @{$f{$d}}
      → prints entire list
  - hashes are really useful for non-numeric data ☆
    (goto for stuff w/dif. characters & useful for sorting)
    - Sorts lexically

---

fgets is a "blocking" call

→ All read/writes are "blocking" calls in linux by default

- Way around, change filemode to non-blocking

- LED tutorial (elinux?); sysfs

* A better gcc (make files!)

=> Caution: most people don't understand <u>make</u>

=> Best source, man & info pages for make

-> Create <u>m</u>akefile

[* To-Do:
- Review navigating man pages on Pi]

-> TARGET = blink //a hash

define => variables

CFLAGS = -g -Wall

OBJS = blink.o //~~compiled bi~~ make will make these!

LIBS = -lm //link math library

{ .f ; fortran
{ .s ; assembler

-> <u>Write Rules:</u>

all: ${TARGET} <u>could list doc. here</u> <- <u>Default rule #1</u>

${TARGET}: ${OBJS}

* <u>\t</u> ↖ Makes the c-compiler go!

-> <u>object</u> files

↳ → ${CC} -o ${TAR...
OBJS
LIBS

-> make isn't good @ linking, need to be explicit

[* make -n
(test it - shows what it <u>would</u> do, test!)]

* indicate tab using:
→

[* Make looks @ existence of file & timestamps!]

-> if there's a 'clean' file... use ".PHONY: all clean"

<u>New Project:</u>

cp it!

-L
↖ searches other directories for <u>libs</u>

* Don't list .c for <u>OBJS</u>!!

Kernel Drivers: Prep

(i.e. TaskManager)

ZOMBIELAND : "Processes"

=> An abstraction of a (running) program that includes system resources: memory, cpu time, (hardware).

- Every process has a unique ID (PID) 1 -> $2^{16}-1$ (where's Ø?)
- Reused. After process dies, PID becomes available-reused

=> #ps agxU -> sysV
#ps -xf... -> BSD
> What are these?

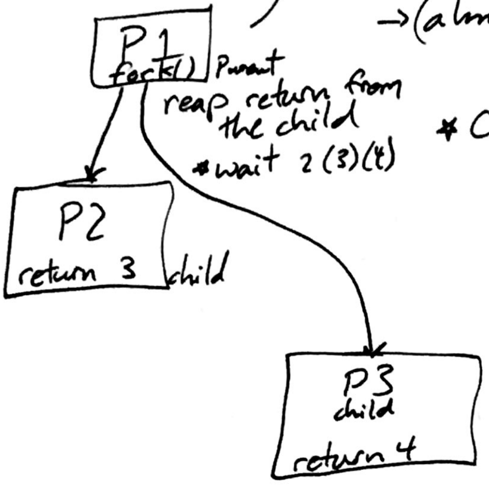
=> Attributes

- UID/GID
- niceness //how willing you are to give up process
- controlling terminal
- signals //send signals to process
  - -> use 'Kill term' to terminate process
    'stop'
    'con't'
    'HUP' //hangup reinitializes it

- "init" is parent of all processes (everything is descendant)

=> States

- R (running), using CPU
- S (sleeping), usually means don't need CPU (woken by interrupt)
- D (waiting on I/O), disk wait
- T (traced), used for debugger
- Z (zombie), can't kill this!



-> fork() makes children
  -> (almost) identical copy

* Check for errors!

* Child dies
  -> Parent gets signal that sick child
  -> Reaps return value from child

  -> if parent dies, child dies w/ no return reaping, ~~be~~ becomes a zombie... ☹
  (return val. wasn't collected)

[=> All it's using is using entry in process table, that's it]