```
<OS&NW@Tutorial-2:~/MS-Teams>echo "Hello! Welcome to Tutorial-2"
Hello! Welcome to Tutorial-2
<OS&NW@Tutorial-2:~/MS-Teams>echo "Learn to build your own FFS"
Learn to build your own FFS
<OS&NW@Tutorial-2:~/MS-Teams>echo "Fully Functional Shell :)"
Fully Functional Shell :)
<OS&NW@Tutorial-2:~/MS-Teams>echo "Let's get started?"
Let's get started?
<OS&NW@Tutorial-2:~/MS-Teams>
```

Points to note

- Assignment 3 is a continuation of this assignment
- Write **modular** code
- When in doubt, mimic shell behaviour
- Include a README
- Don’t copy!!

Processes & getpid()

- An instance of a running program is called a **process**
- Each process gets a unique **process ID** or pid (a 5 digit ID number in unix / linux).
- **getpid()** function returns the process ID of the calling process

fork()

- System call used for creating a new process - **child process**
- Both processes execute the next instructions after fork()

What does it return?

- **Negative Value**: creation of a child process was unsuccessful :(
- **Zero**: Returned to the newly created child process
- **Positive value**: returned to parent or caller. The value contains **process ID of newly created child process**.

Use the returned value to conditionally execute commands

Google

Killing a child

Google Search I'm Feeling Lucky

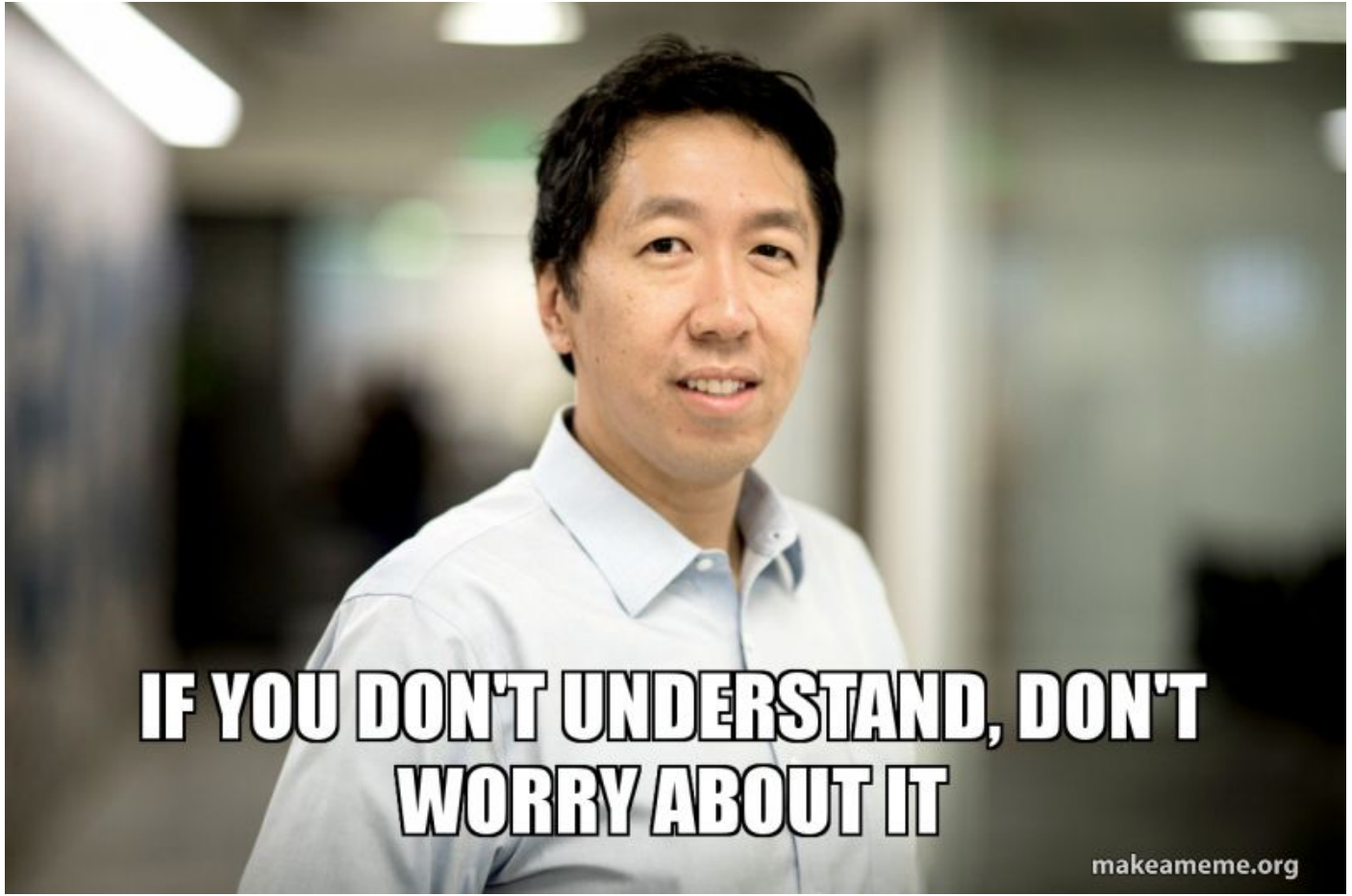
Google

Killing a child process stackoverflow

Google Search I'm Feeling Lucky

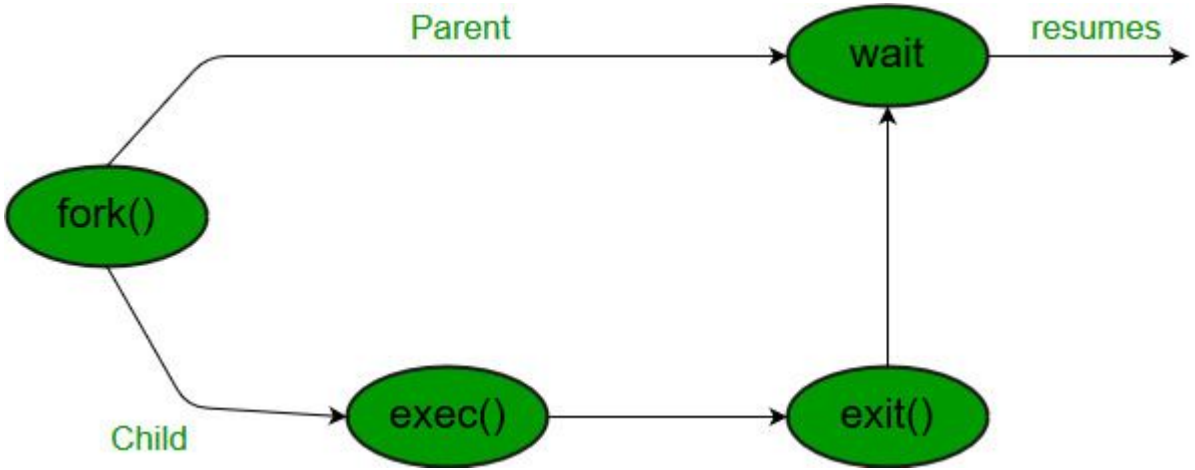
Oh, man wait

- System call that waits for state changes in a child of the calling process
- Example: child terminated; the child was stopped by a signal; or the child was resumed by a signal
- **waitpid**() system call suspends execution until a child specified by *pid changes state*
- man page for different types of usage


IF YOU DON'T UNDERSTAND, DON'T WORRY ABOUT IT
makeameme.org

execvp()

- Replaces the current running process with a new process
- **exec** type system calls allow a process to run any program files



Example of fork, exec, and wait together

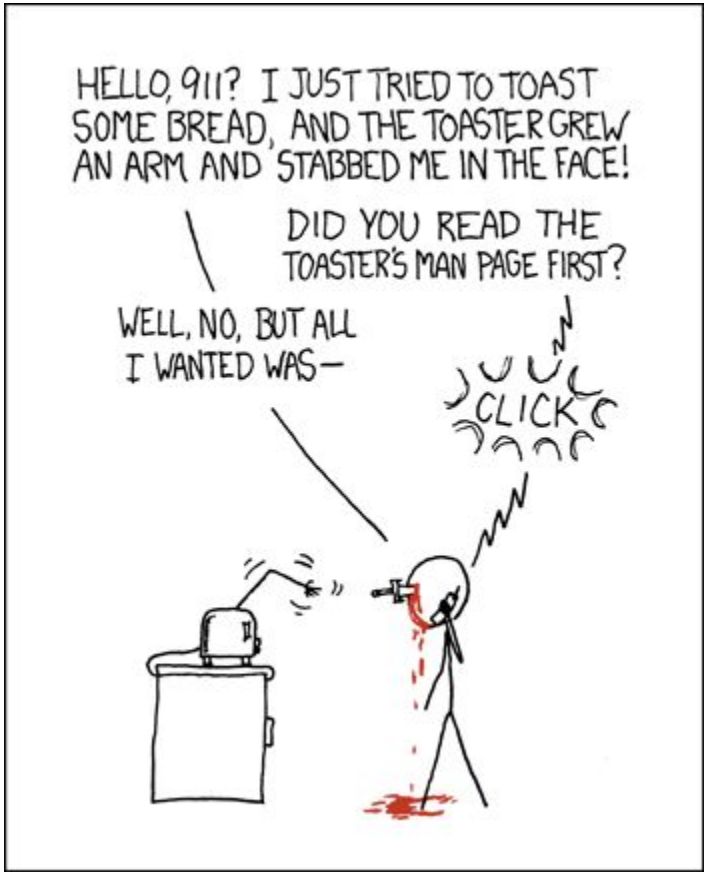
Reaction to assignment:

How to start?

Useful commands/structs/files

- uname
- hostname
- signal
- waitpid
- getpid
- kill
- execvp
- strtok
- fork
- getopt

- readdir
- opendir
- closedir
- sleep
- watch
- struct stat
- struct dirent
- /proc/interrupts
- /proc/loadavg
- man pages :)


HELLO, 911? I JUST TRIED TO TOAST SOME BREAD, AND THE TOASTER GREW AN ARM AND STABBED ME IN THE FACE!
DID YOU READ THE TOASTER'S MAN PAGE FIRST?
WELL, NO, BUT ALL I WANTED WAS—
CLICK

Any doubts?