Homethink Questions

What do the following lines of code mean? Sketch a picture of your answer if you can!

1. Looking at C code, you may find a lot of examples of these parameters, what do they mean?

```
int main(int argc, char **argv)
```

they are command-line arguments passed to main func,

argc(argument count) counts the number of command-line arguments passed to main func, the number would be 1(represent the file path itself) and the number of   char array to execute

argv(argument vector), the char array to run parameters where each char* represent the path of program to execute and parameter passed into

2. Vido formatted his code differently, but what does this mean?

```
char *line;
```

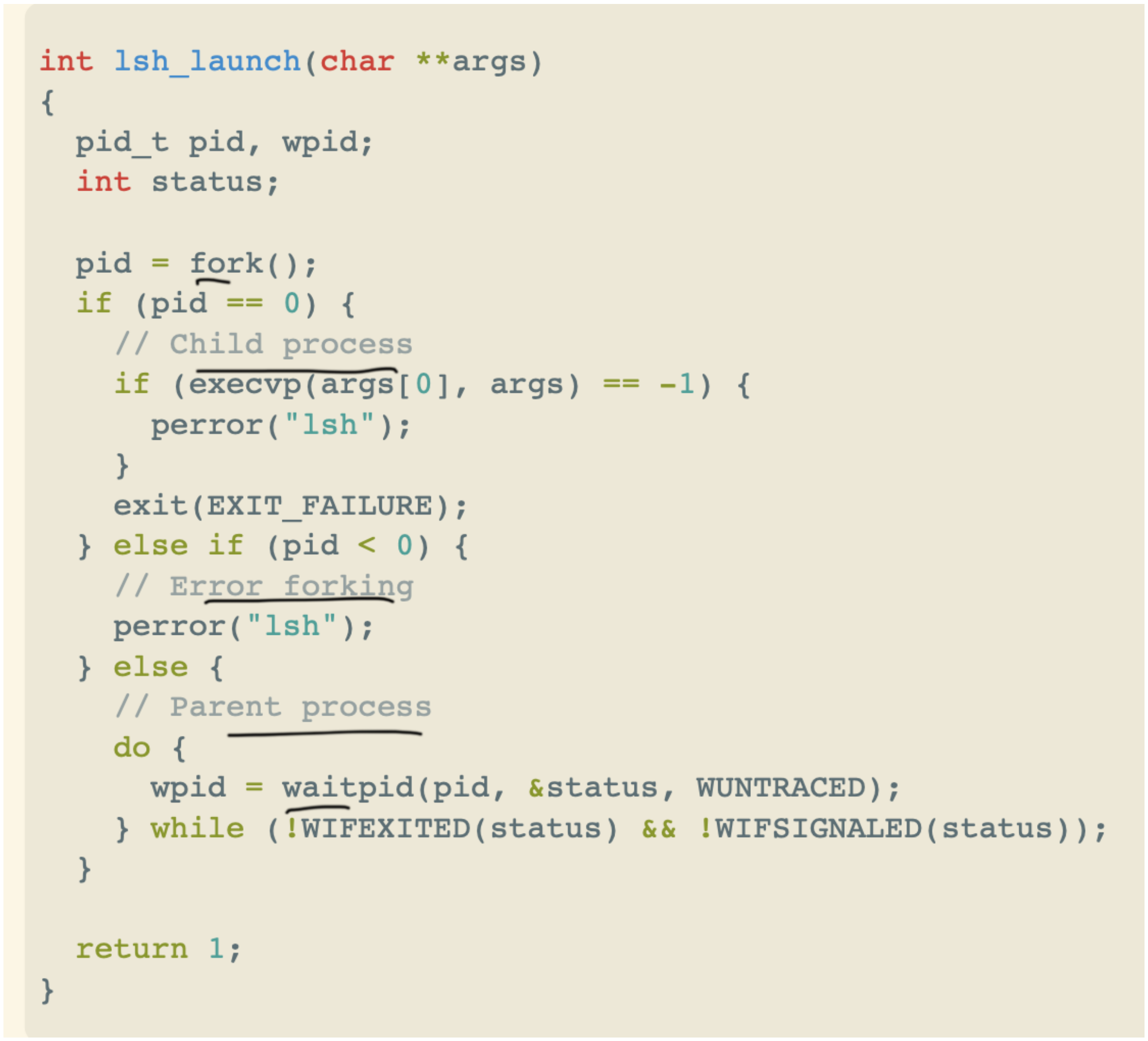
declare a pointer variable line pointing to char variable, used to store command-line inputs

3. Malloc! Wow, what is this all about?

```
char *buffer = malloc(sizeof(char) * bufsize);
```

allocate continuous bytes of memory to store buffersize and return as a pointer called buffer as an array, which makes traversal more easily

4. What do fork() and execvp() do in the following code? What about waitpid()?



```c
int lsh_launch(char **args)
{
  pid_t pid, wpid;
  int status;

  pid = fork();
  if (pid == 0) {
    // Child process
    if (execvp(args[0], args) == -1) {
      perror("lsh");
    }
    exit(EXIT_FAILURE);
  } else if (pid < 0) {
    // Error forking
    perror("lsh");
  } else {
    // Parent process
    do {
      wpid = waitpid(pid, &status, WUNTRACED);
    } while (!WIFEXITED(status) && !WIFSIGNALED(status));
  }

  return 1;
}
```

fork() start a process, when called, os duplicates the process then start both processes(original process is parent, new one is child). fort return 0 to child process and return the process ID number of child to its parent; to conclude, fork start a new process by duplicating an existing one

execvp() takes two parameters, the first argument is the file, the second is an array of instructions to execute, it check if child process receive specific instructions, if not(-1), it print the error message(perror)

waitpid() suspends the calling process till a child process with given pid changes its state, the third argument is the option to modify its behavior, in this case, WUNTRACED return if child process stops, and provide the status of traced stopped child

Reference:

https://linux.die.net/man/2/waitpid

WIFEXITED(*status***)**

returns true if the child terminated normally, that is, by calling **exit**(3) or **_exit**(2), or by returning from main().

WIFSIGNALED(*status***)**

returns true if the child process was terminated by a signal.