CS241 Lecture 8 Angrave – Forking Waiting and Signals

Q1 My long-running server process creates many child processes and never calls wait() or waitpid(). What have I created?

Q2 What is a signal? How do they work?

Q3 How do I send signals programmatically to a process?

Q4 How do I send a user-defined signal? Terminate signal?

Q5 Why should I use the signal symbols not the constants?

Q6 What is the alarm signal?

int main() {

int x = 0;

while(x++ < 60) {

char \*mesg = "." ;

write(1, \_\_\_\_ , \_\_\_\_\_\_ );

sleep(1);

}

return x;

}

WIFSIGNALED, WTERMSIG, WEXITSTATUS, WIFEXITED,\_ SIGALRM, perror,fork,waitpid

pid\_t childid;

void child() {

\_\_\_\_\_\_ Make the alarm go in 1 second

\_\_\_\_\_\_\_ Sleep for 2 seconds

puts("I'm the child exiting normally");

\_\_\_\_\_\_\_ exit normally

}

void parent() {

int status;

\_\_\_\_\_\_\_(childid, &status, 0);

if(\_\_\_\_\_\_\_\_\_\_ (status)) {

printf("Child exited due to signal %d\n", \_\_\_\_\_\_(status));

if((\_\_\_\_\_\_\_\_\_\_ (status) == \_\_\_\_\_\_\_\_\_)

puts("ALARM CLOCK");

}else

if( \_\_\_\_\_\_\_(status)) {

printf("Child exited with %d\n", \_\_\_\_\_\_\_\_\_\_\_(status));

}

}

int main() {

printf("Hello world!\n");

childid = \_\_\_\_\_\_\_\_\_ ();

if(childid >0) parent();

else if(childid == 0) child();

else \_\_\_\_\_\_\_\_\_ ("fork failed"); // print error

return 0;

}

If I started my program with "./program file.txt" how would I read the the first argument (file.txt)?

What is special about value of argv[0]?

What is the value of argv[argc]?

What is a shell? How would a shell print "Seg fault" or "Alarm clock?"

What is wrong with the following program that tries to implement a shell?

char buffer[1024];

puts("Enter Your command oh master and I will exec it. Lol catz")

while(1) {

fgets(buffer, sizeof(buffer), stdin);

char\* args[]= {".",NULL};

execvp(buffer, args);

}

}

How would you implement background commands? ("gcc ... &")

How would you implement reading a script from a file?

How would you implement redirecting output of a command to a file?

e.g. cat file.c > output.txt

What is "POSIX"?

Give some examples of the security features of an operating system

What is a fork bomb()?

What is a heap allocator? Is it part of a process or part of the operating system? What must it do efficiently?

What are the advantages of this allocator? disadvantages?

void\* malloc(unsigned int numbytes) {

// sbrk increases the process's data segment by n bytes

void\* ptr = sbrk(numbytes);

if(ptr == (void\*) -1) return NULL; // no memory for you!

return ptr;

}

void free(void\*mem) { /\* do nothing \*/}