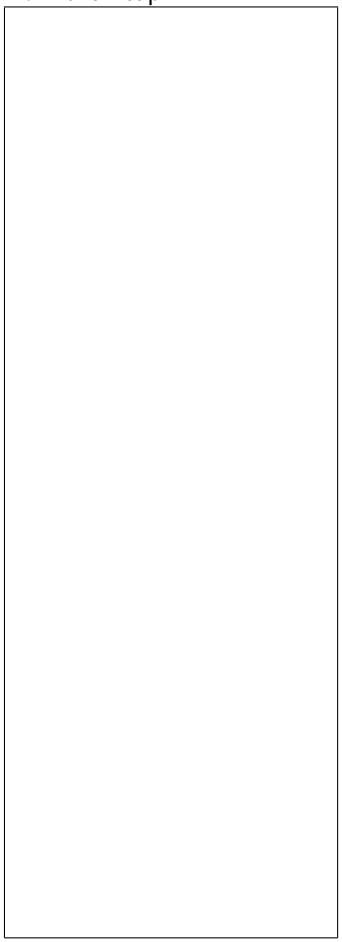
CS241 - Processes

This week we are going to be dealing with processes and the fork-exec-wait model.

Address Space

Using the box below, draw the address space of your typical process (remember the stack, the heap, the text segment, the data segment). Label which parts are writable and which are executable. Is the stack far away from the heap?



Cloning

Without looking at your notes, list as many things as you can remember that are copied over when your process forks. What isn't?

-

Where is the subtle forkbomb in this code?

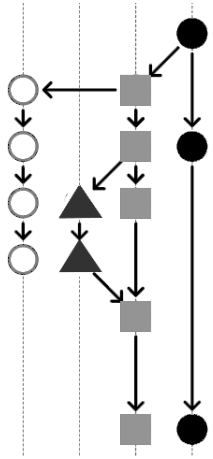
```
#define PROC 10
int main(){
    pid_t processes[PROC];
    for(int i = 0; i < PROC; ++i){
        processes[i] = fork();
        if(!processes[i]){
            execlp("ruby", "ruby", "file.rb",
                (char*)NULL);
        }
    }
    for(int i = 0; i < PROC; ++i){
        waitpid(processes[i], NULL, 0);
    }
}
```

Process Flowchart

Let's draw a process flowchart! Here is an example

```c
int main(){
    pid_t child1 = fork();
    if(!child1){
        pid_t child2 = fork();
        if(!child2){
            while(1) {}
        }
        pid_t child3 = fork();
        if(!child3){
            printf("Hello!");
            exit(2);
        }
        waitpid(child3, NULL, 0);
        waitpid(child2, NULL, 0);
        exit(1);
    }
    waitpid(child1, NULL, 0);
}
```

May look like this (the rightmost process may be the first one created)



```c
int main(int c, char **v){
    while (--c > 1 && !fork());
    sleep(c = atoi(v[c]));
    printf("%d\n", c);
    wait(NULL);
    return 0;
}
```

```
$ ./main 2 3 4 1
```