From /cs/faculty/rich/cs170/test_execs/exec.c:

```c
char *path = "argtest";
printf("printf 0x%x, 0x%x %s\n", argv, argv[0], argv[0]);
execve(path, argv);
```

→ Header for execv:
```c
int execv(const char *pathname, char *const argv[]);
```

→ Your header for implementing execv:
```c
void do_execv(pcb *proc);
```

Execv Practice!

1. What is the type of path? What is the type of proc->registers[5]?
   ```
   char* int → char*
   ```

2. What is the type of argv? What is the type of proc->registers[6]?
   ```
   char*, char* [6] int → char**
   ```

   a. What is the value of path (path is a pointer, so it is an int)? 80
   b. Where is the value of path really located (this is an int)? 120

4. Also let proc->registers[6] = 72
   a. What is the value of argv? 72
   b. What is the type of argv[0]? char*
   c. Suppose argv[1] = 96, where is "Once" really located? 0 + 40 + 96 = 136
   d. What is the value of argv[9]? 0x0
Fork Practice!

1. How many processes return from a fork() system call?

2. The following code for do_fork() is wrong! What’s wrong with it?

```c
void syscall_return(pcb *proc, int ret){
    proc->reg[PCReg] = proc->reg[NextPCReg];
    proc->reg[2] = ret;
    queue_proc(proc);
    kt_exit();
}

void do_fork(pcb *proc){
    pcb *child = (pcb *)malloc(sizeof(pcb));
    child->pid = parent->pid;
    child->base = parent->base;
    child->limit = parent->limit;
    for (int i = 0; i < NumTotalRegs; i++)
        child->registers[i] = parent->registers[i];
    memcpy(child->base, proc->base, proc->limit);
    syscall_return(child, 0);
    syscall_return(parent, child->pid);
}
```
Wait Practice!

1. What is a zombie? Who cleans up after a zombie?
2. What is an orphan? Who cleans up after an orphan?
3. Consider the following:

Init is initialized (a process to continually wait on its children to exit)
   → Who is Init’s parent? Nobody

proc1 is initialized
   → Who is proc1’s parent? Init

proc1 calls fork(), so proc2 is initialized
   → Who is proc2’s parent? proc1

proc2 calls fork(), so proc3 is initialized
   → Who is proc3’s parent? proc2

proc2 calls exit()
   → Does anyone clean up proc2? no
   → Who is proc3’s parent? Init

proc1 calls wait()
   → Who does proc1 clean up in this call? proc2

proc1 calls exit()
   → Who cleans up proc1? Init
   → Who is proc3’s parent? Init

proc3 calls exit()
   → Who cleans up proc3? Init

all processes except Init have exited
   → Who cleans up Init? OS