CSE 451: Section 2 Handout
4/11/2019

Open File Table and Intro to Multi-Processing

Open File Table:

What does it mean for two entries in the process open file table to point at the same global open file table entry?

Draw out the process and global open file table layout after the following c code:

```
int fd1 = open("file.txt", O_RDONLY);
int fd2 = open("file.txt", O_RDWR);
```

These are the first few lines of lab1test.c. What is this doing?

```
open("console", O_RDWR);
dup(0);
dup(0);
```

Linux supports more functions built on top of dup, `dup2` is an example.
Take a look at the man pages for `dup2` and discuss how you would implement them in xk.

CSE 451: Section 2 Handout
4/11/2019

Open File Table and Intro to Multi-Processing

Multi-Processing:

Take a look at `struct proc` in proc.h and the `allocproc` function in proc.c.
When a new process is created with `fork()`, what should the `struct proc` values be set to?

Where will the new process start executing in user mode? (and where is this information stored?)

To differentiate the new processes from the old process, we call the new process a child of the parent old process. The return value of `fork` is different between the child and the parent. The parent will return the current process id and the child will return 0.
How can we alter the return value of the fork function to simulate the situation above?

`exit` and `wait` are two important synchronization methods in xk. `wait` waits until a child has exited and returns the process id of the child. `exit` exits the current process.

Why can a process not clean itself up on `exit`?
If a process cannot clean itself, when are a processes data structures free'd?