Lecture 3 (Threads, Address Spaces, Filesystem, Devices)

1 More (Recap?) about Processes

1. Address space of the process is restricted to the process
2. OS on performing `init` creates some processes which create some other processes in return
3. OS cleans up entire data of child process only after wait is called since exit code of child is needed (such child processes are called zombie processes)

2 Signals

1. Signals raised by OS are sent to signal handler created by process (else default is used)
2. Such a handler interrupts the execution of the process and executes its tasks
3. Execution resumes after handler returns
4. It is like an async function call

2.1 Syntax

```
signal(int signum, void (*handler)(int))
```

2.2 Standard Signals

1. SIGINT
2. SIGSTOP (cannot be overwritten)
3. SIGSEGV
4. SIGFPE
5. SIGCHLD (child exits, helps in preventing zombie processes when commands are run as `ls &`, i.e., when parent doesn't wait)

2.3 `kill`

A command used to send signals to other processes

```
kill(pid, signum)
```

3 Pseudo-Filesystem

`/proc/` is a pseudo-fs which contains information about processes (each pid has its own directory), this directory can be read/written using open/close

4 Threads

1. `write`, `read` are heavy calls and if processes require many read-write operations, it is not suggested to use these syscalls
2. Instead concept of threading can be used
3. Threads are part of processes which share the same address space but run independently
4. There are two ways of creating threads:
    - Kernel-level: having a syscall
    - User-level: creating 'logical' threads, which don't execute concurrently (when blocking via some syscalls), but only logically - a scheduler is needed