Lecture 1 (Introduction to UNIX System Calls)

1 History of OS

1. The very first OS was a set of device drivers, it was a uni-process OS
2. UNIX was a breakthrough OS developed by Ken Thompson along with Dennis Ritchie etc

2 UNIX Properties

1. Memory was exclusive but data was shared in UNIX
2. Filesystem was developed
3. Shell was implemented inside the kernel
4. Kernel functions are called system calls
5. Shared resources are mediated by the OS (hardware devices)

2.1 Reading from File

1. Filenames were used with directory structure
2. Concept of file descriptor existed (fd)
3. `read/write(fd, buffer, #bytes)` was the way to read/write to file

2.2 Resource Usage

1. They are also used similar to reading from file - `open`, `read`, `write`, `close`
2. Errors are returned for invalid usage

2.3 Shell Program

1. It is the base program which transfers control to executables based on user commands
2. Exiting the program returned back to the shell
3. Opened files were closed on exiting

2.3.1 Improvements from the Original UNIX Version

1. Shell is no more a part of OS and is made an application, the kernel has been minimised
2. UNIX provides syscalls `fork` and `exec`

- `exec` replaces current program with the new program
- `fork` creates identical programs with identical states and program counters
  i. Parent process is stored to disk since only one process can run at any time
  ii. Child process calls `exec`
  iii. `fork` returns the *pid* of child for parent and 0 for child

2.3.2 Basic Shell Command

```
while(1) {
  readCommand(); // calls other system commands
  pid = fork(); // syscall
  if (pid == 0)
    exec(command);
}
```

2.4 File Descriptors

1. On opening a new 'file', the first empty index from **File Descriptor Table** is returned (this table is hidden from the program)
2. Indices 0, 1, 2 are reserved for STDIN, STDOUT, STDERR respectively