**PROJECT PROPOSAL**



**Fall 2020**

**CSE302L System Programming Lab**

Submitted by:

**Shah Raza** (18PWCSE1658)

**Jamshid Ali** (18PWCSE1654)

**Abdul Mohiz Khalid** (18PWCSE1618)

**Muzamal Ali** (18PWCSE1665)

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr. Madiha Sher**

Saturday, February 13th, 2021

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**SnakeOS**

**Introduction:**

An **Operating System** (OS) is an interface between a computer user and computer hardware. An **operating system** is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

In our Project we will build a multitasking operating system and kernel with an interactive shell from Scratch.

**Programming Languages:**

* C Programming language
* Assembly language

**What to Expect:**

We will try to create a 32-bit multi-tasking kernel that has the FAT16 file system. Our kernel will use Intel's built in memory protection and security mechanisms that allow us to instruct the processor to protect our kernel and prevent user programs from damaging it.  Take advantage of clever instructions in Intel processors to allow all processes to share the same memory addresses, this is known as memory virtualization. We map memory addresses to point to different physical memory addresses to create the illusion that every process that is running is loaded at the same address. This is a very common technique in kernel development and is also how swap files work (Those files that are used to compensate for when you run out of usable RAM). Create our own virtual file system layer that uses a design that is similar to the Linux kernel.

Implement our very own FAT16 file system driver allowing files to be born! Implement functionality for tasks and processes and write our own keyboard drivers. Implement the "malloc" and "free" functions creating our very own heap that's designed to keep track of what memory is being used. Memory management is essential in any operating system and kernel.

Create an ELF file loader, we will compile all our operating systems programs into ELF files and allow the loading of binary programs or ELF programs. ELF files contain a lot of information that describes our program for example where our program should be loaded into memory and the different sections of the program.

So in the end you can expect a fully functioning 32-bit multi-tasking kernel that can have many processes and tasks running at the same time. It will have a working shell that we can use as well.