OS CS219 Notes

January 9, 2024

1 Introduction

An **OS**(operating system) alias **Master Control Program** alias **System software** is software than enables the user to access the hardware resources of a computer in a controlled manner. It acts as a layer of abstraction allowing the user to not worry about hardware level access and just provides access to a few methods needed by the user

The OS has several uses/functions in a computer

- Manages resources as a single **central entity** and hence efficiently
- **Virtualizes** physical resources to be utilized by multiple processes[1]
- **Isolates** and protects processes from one another by not allowing direct access to hardware
- Provides a set of system calls for the user to access hardware resources
- Starts processes and allocates and manages memory required by said process during execution

Thus it is easy to see that the OS has several important functions to perform inorder to enable an abstraction of hardware from the users

2 Process and Virtualization

A process is just the sequence of execution of instructions by the CPU . When we write and compile a program it gets converted into a sequence of instructions whose first *instruction address* is fed to the PC and as we know the sequence of instructions for that program starts getting executed one by one. This is called a process. So when we run a program a *process* is created.

[1] process is defined later

2.1 Context switching

How do we tell the CPU to start a process. First obviously we need to feed the address of the first instruction of the process to the **Program Counter**. We also need to set the stack pointer and other registers with appropriate values. This is called **setting up the context** for a process. This job is done by the OS .After the context is set the OS takes a back seat and allows the CPU to do its work

However this has a few issues. Firstly, when the process requests for data from say the hard drive there is a gap where the CPU is on standby which is wasteful since other processes also require it. Secondly, we also want responsiveness from our system ie when we have a process running we also may want to interact with other processes.

Both of these issues are solved by **concurrent** execution. Basically we run a process for a while and when the CPU is on standby or after a particular interval of time we save the *context* (ie) states of all registers including PC and start the other process by setting up its context this repeats for a while and eventually the partially executed process's context is set and it is continued. This is reffered to as **context switching** and is an important part of Virtualization[2] of the CPU. Note that a part of the OS (ie) *OS scheduler* decides which program to run at what time

2.2 Virtualization

Virtualization refers to the creation of an illusion that each of the processes have full access hardware resources[3]. This enables the hardware to act much more powerfully than they are capable of. For example, as mentioned above context switching creates the illusion of the presence of multiple cores each assigned to one process whereas in reality it is just one core. Infact this is refered to as **Hyperthreading**. Apart from the CPU memory, addresses can be virtualized.

3 Memory allocation and Isolation

3.1 Memory

As we learnt in CS230, memory for a process/program is allocated as a fixed number of bytes. The inital bytes of this memory is the instructions and the global/static variables. Local variables aren't initialised in memory since we do not know the number of times each function is called, instead we have a dynamically growing stack whose starting

[2]Read further to know what it is

[3]It is useful to think of Virtualization as the OS lying to each process about it having full access to a resource

address is stored in the special *stack pointer register*. This stack grows and shrinks as necessary functions are called and they return values. [4].

Apart from this we have a heap which can be accessed by user to store dynamically increasing data structures. We can request the OS to allocate certain number of bytes and return a pointer to said bytes

Here again however the OS plays tricks on the processes. Since it is impractical for the OS to allocate to allocate memory for the process continguosly it allocates them in chunks but returns a **virtual address** (Recall Virtualization) to the program. This "virtual address" is the address returned when the user requests the OS for an address of the data stored. Here again the OS lies to the process creating the illusion that it has access to continguos memory starting from some location

3.2 Isolation

Now we understand that the OS allows multiple processes to run at the same time and share resources But this raises a huge issue since processes are supposed to be independent and processes being to affect other processes would cause problems. The OS takes care of this too by maintaing strict control over access to hardware

The OS is the only entity with access to hardware and process can make specific requests to the OS to use hardware via *system calls*[5]. Infact there are two types of instructions and processer modes corresponding to them

- **Privileged instructions** - special instructions that can interact with hardware. Generated by syscalls,device drivers CPU is in **kernel** mode while executing them
- **Unprivileged instructions** - simple instructions that do not need access to hardware. Given by user processes. CPU is in **user** mode while running them

The CPU is always in user mode except the following cases.

- A syscall is made
- Interrupt occurs
- Error needs to be handled

Interrupt: In addition to running programs a CPU has to handle external inputs from devices like a mouse click. This is called an Interrupt. During an interrupt control is given to the OS(Kernel mode of CPU) which deals with the interrupt and returns control to the processes[6]

[4]The structure used here is a stack since functions are inherently *LIFO* functions called last return first

[5]syscalls can't be accessed directly usually. They are in a language's standard library

[6]This means saving context, handling the interrupt and setting context of the past status