

Task 2

parts of core os and how it works

There are following 8-components of an Operating System:

1. Process Management
2. I/O Device Management
3. File Management
4. Network Management
5. Main Memory Management
6. Secondary Storage Management
7. Security Management
8. Command Interpreter System

Functions & features of operating system in a computer

An operating system has lot of functions to do in a computer. Some of them are listed below:

1) Provide user interface - As said earlier without a proper user interface its difficult to manage a machine. It is possible to interact with a machine using the commands, but it's easy to learn all the commands used in an OS.

2) Input/output management - We can add additional hardware's to a computer and configure them easily with the help of an operating system. Printer, external hard disk, scanner, USB drives are some of the external devices we can connect with a computer.

3) Memory and CPU management - Operating system allocates the memory and other resources for the other programs in a computer. The memory allocation and CPU usage of each program is controlled by OS and you can check them from the 'Task Manager'.

4) Multitasking - Multitasking is another feature of operating systems (not in DOS). We can do more than one tasks at the same time in an operating system. For example you can listen to music in media player while typing a note.

5) Networking - Its easy to do networking with the help of operating systems. You can share files, folders or even hardware resources like printer through a network with the help of operating systems.

Kernel is central component of an operating system that manages operations of computer and hardware. It basically manages operations of memory and CPU time. It is core component of an operating system. Kernel acts as a bridge between applications and data processing performed at hardware level using inter-process communication and system calls.

Kernel loads first into memory when an operating system is loaded and remains into memory until operating system is shut down again. It is responsible for various tasks such as disk management, task management, and memory management.

It decides which process should be allocated to processor to execute and which process should be kept in main memory to execute. It basically acts as an interface between user applications and hardware. The major aim of kernel is to manage communication between software i.e. user-level applications and hardware i.e., CPU and disk memory.

Objectives of Kernel :

- To establish communication between user level application and hardware.
- To decide state of incoming processes.
- To control disk management.
- To control memory management.
- To control task management.

Types of operating systems

Operating systems are classified into different types they are:

1) Single user, single tasking operating system - Single user single tasking operating systems are those operating systems which can be used by a single user at a time. Also we can only do a single task at a time in such operating systems. Example of single user single tasking operating system is **MS DOS**. In Microsoft DOS only one program can be executed by a user at a time.

2) Single user multi tasking operating system - Single user multi tasking operating systems are those operating systems which can be used by a single user to do multiple operations/tasks at a time. Microsoft Windows is an example of such operating system. In Windows only one user can log in at a time but that user can do many tasks at a time.

3) Multi user, multi tasking - In multi user multi tasking operating systems more than one user can log in at a time and do as many tasks they want. Linux is an example of such operating system. This is the main [difference between a windows and Linux](#) operating system. In Linux

there are 7 terminals, in which 6 are non graphical terminals and 1 is graphical terminal. So 7 users can use a Linux machine at a time.

4) **Real time operating systems** - Real time operating systems are operating systems used in real time applications like embedded systems, robots, automobile engine controllers etc. Windows CE is an example for real time operating system.

5) **Chrome OS** - Chrome OS is the new operating system developed by Google and it cannot be added in any of the classification listed above. It's not actually an operating system but its a web browser with advanced feature. Imagine a computer only having an internet connection and Chrome OS and that's what Google is making. In a computer with Chrome OS the data will be stored in internet and anything we do or install in an OS will be done in internet space. So its sure that Chrome OS is going to make a revolutionary change in the history of operating systems.

▪ Types of language we give to machines

Two Basic Types of Computer Language

- **Low-Level Languages:** A language that corresponds directly to a specific machine
- **High-Level Languages:** Any language that is independent of the machine

There are also other types of languages, which include

- **System languages:** These are designed for low-level tasks, like memory and process management
- **Scripting languages:** These tend to be high-level and very powerful
- **Domain-specific languages:** These are only used in very specific contexts
- **Visual languages:** Languages that are not text-based
- **Esoteric languages:** Languages that are jokes or are not intended for serious use

```
In [1]: x=input("please enter your username:")  
please enter your username:Farah
```

```
In [*]: import getpass  
pswd = getpass.getpass('Password:')  
Password: *****
```

```
In [ ]:
```

```
In [1]: x=input("please enter your username:")  
please enter your username:Farah
```

```
In [*]: import getpass  
pswd = getpass.getpass('Password:')  
Password: farah
```

```
In [ ]:
```

10 tips for writing cleaner code in any programming language

1. Use descriptive names. ...
2. Use empty lines to create a readable code. ...
3. Do not send more than three parameters into a function. ...
4. Remember the functions must do only one thing. ...
5. Functions must be small. ...
6. Reduce the number of characters in a line. ...
7. Avoid using comments.

localhost:8890/notebooks/Orange/Untitled1.ipynb?kernel_name=python3

jupyter Untitled1 Last Checkpoint: 15 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [4]: #do {
#statement
#} while (condition);

In [5]: i = 1

while True:
    print(i)
    i = i + 1
    if(i > 5):
        break

1
2
3
4
5

In [11]: for i in range(1, 8):
    print(i, i * 2)

1 2
2 4
3 6
4 8
5 10
6 12
7 14

In [ ]:
```

Activate Windows
Go to Settings to activate Windows.

91°F Sunny 10:29 PM 9/1/2022

localhost:8890/notebooks/Orange/Untitled1.ipynb?kernel_name=python3

jupyter Untitled1 Last Checkpoint: 11 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
if word != secret_word and counter > 7:
    break

Enter the secret word: m
Enter the secret word: ,
Enter the secret word:
Enter the secret word: m
Enter the secret word:
Enter the secret word: m
Enter the secret word: m
Enter the secret word:

In [4]: #do {
#statement
#} while (condition);

In [5]: i = 1

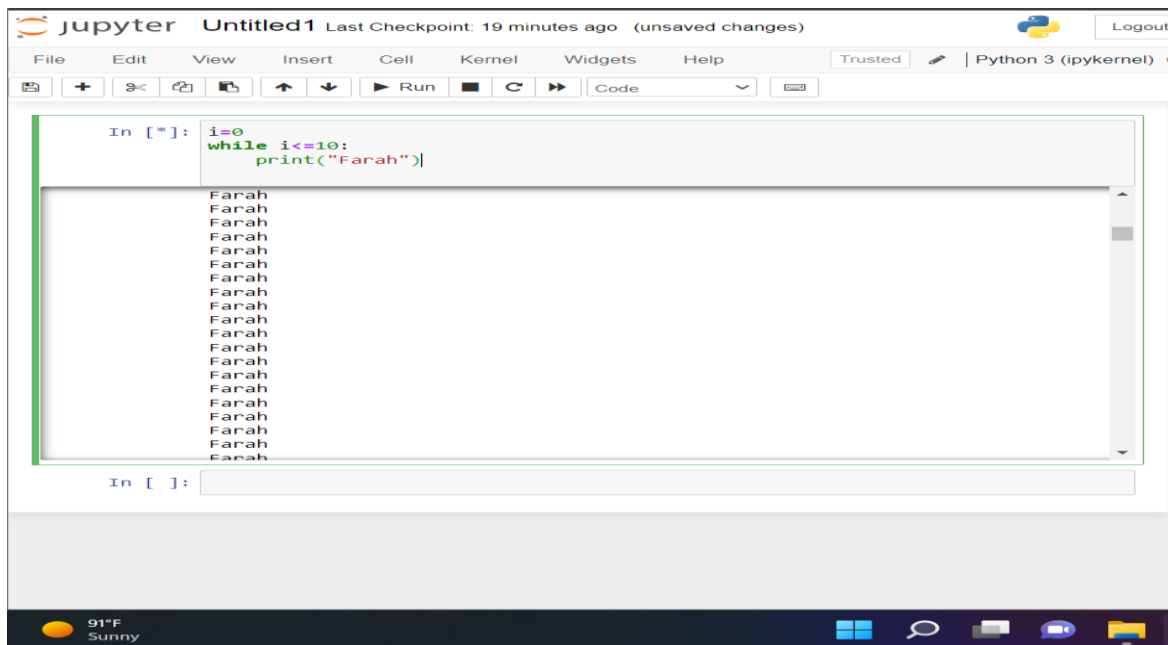
while True:
    print(i)
    i = i + 1
    if(i > 5):
        break

1
2
3
4
5

In [ ]:
```

Activate Windows
Go to Settings to activate Windows.

91°F Sunny 10:25 PM 9/1/2022

A screenshot of a Jupyter Notebook interface. The top bar shows 'jupyter' and 'Untitled1' with a 'Last Checkpoint: 19 minutes ago (unsaved changes)' status. The right side has a 'Logout' button. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. Below the menu is a toolbar with icons for file operations and execution. The main area contains a code cell with the following Python code:

```
In [*]: i=0
while i<=10:
    print("Farah")
```

The output of the code is displayed below the code cell, showing the word 'Farah' printed 10 times on separate lines. The bottom status bar shows '91°F Sunny' and a Windows taskbar with various icons.

In object-oriented programming ([OOP](#)) software design, dependency injection (DI) is the process of supplying a resource that a given piece of code requires. The required resource, which is often a component of the application itself, is called a dependency.

When a software component depends upon other resources to complete its intended purpose, it needs to know which resources it needs to communicate with, where to locate them and how to communicate with them.

One way of structuring the code is to map the location of each required resource. Another way is to use [dependency injections](#) and have an external piece of code assume the responsibility of locating the resources. Typically, the external piece of code is implemented by using a framework, such as [Spring](#) for Java applications.

Threads

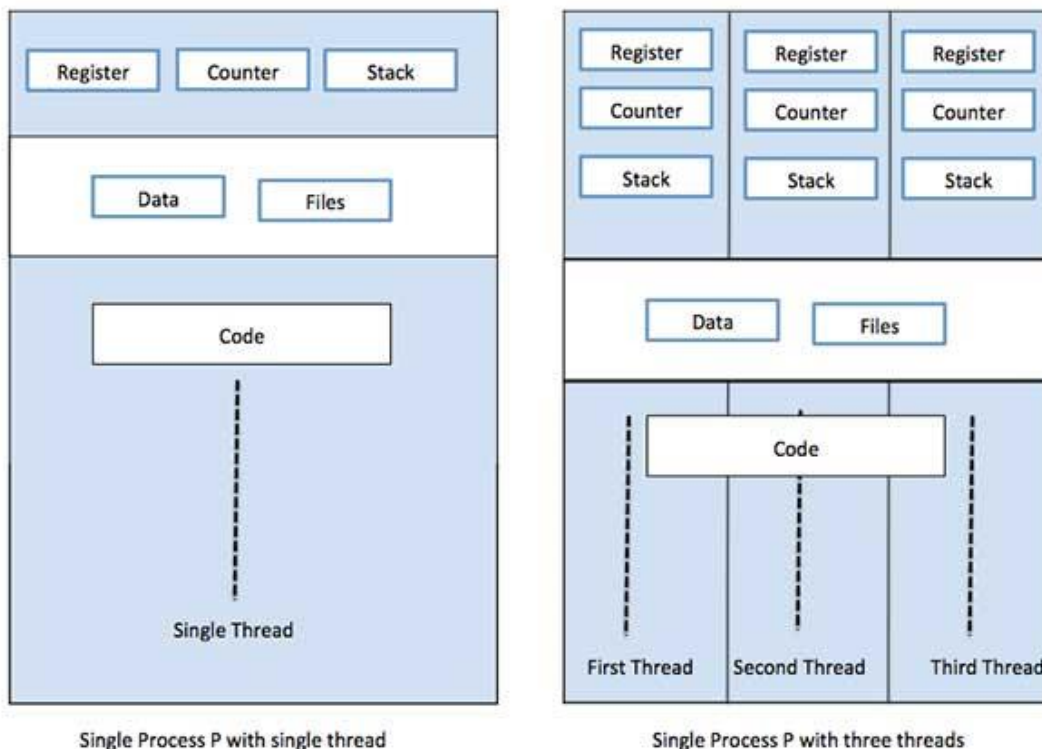
What is Thread?

A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

A thread shares with its peer threads few information like code segment, data segment and open files. When one thread alters a code segment memory item, all other threads see that.

A thread is also called a **lightweight process**. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. The following figure shows the working of a single-threaded and a multithreaded process.



Difference between Process and Thread

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Advantages of Thread

- Threads minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to create and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

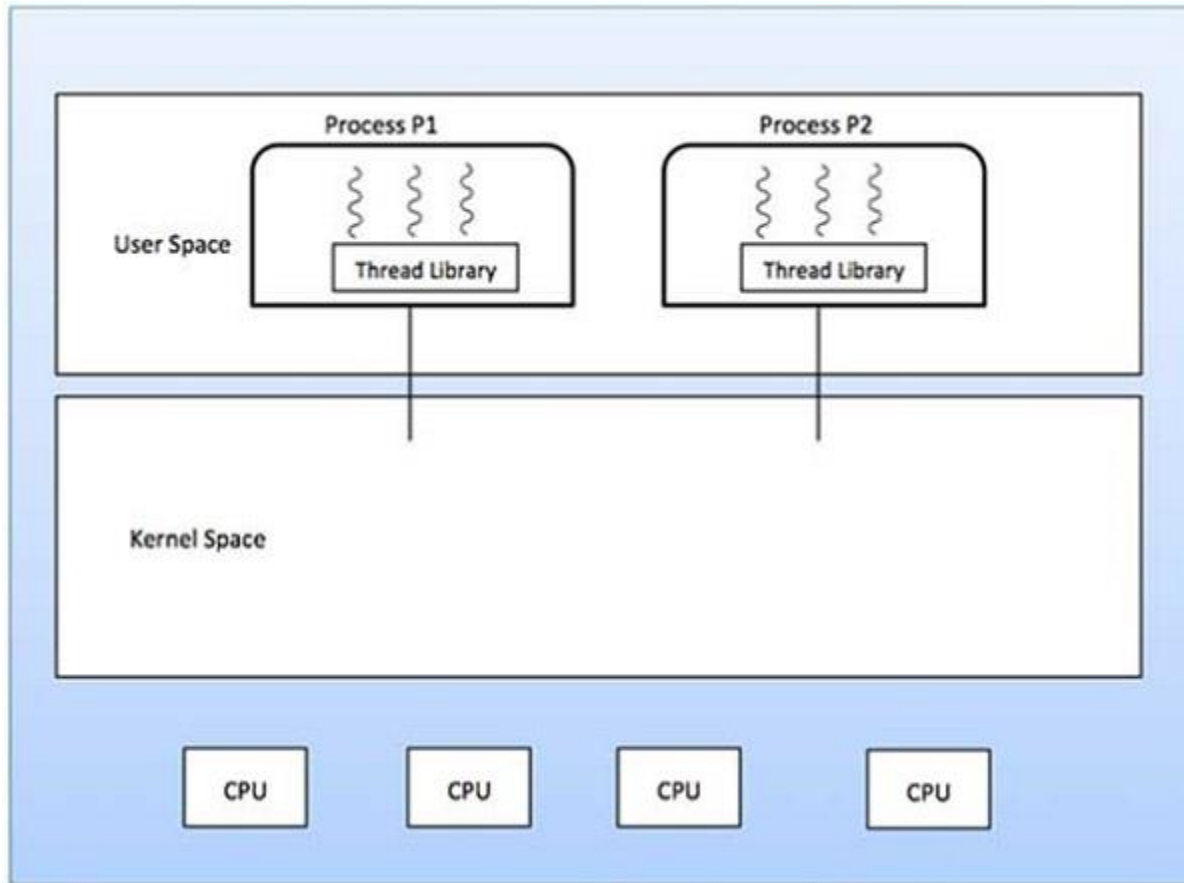
Types of Thread

Threads are implemented in following two ways –

- **User Level Threads** – User managed threads.
- **Kernel Level Threads** – Operating System managed threads acting on kernel, an operating system core.

User Level Threads

In this case, the thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.



Advantages

- Thread switching does not require Kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.

Disadvantages

- In a typical operating system, most system calls are blocking.
- Multithreaded application cannot take advantage of multiprocessing.

Kernel Level Threads

In this case, thread management is done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

The Kernel maintains context information for the process as a whole and for individuals threads within the process. Scheduling by the Kernel is done on a thread basis. The

Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.

Advantages

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
- Kernel routines themselves can be multithreaded.

Disadvantages

- Kernel threads are generally slower to create and manage than the user threads.
- Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.