**Process concepts**

State of execution

Program counter

Stack

Parts and temporary holding area

Data, register state,

Occupies state in memory

May require special hardware

I/O devices

What is a process - OS manages hardware on behalf of applications

The process is a state of a program when executing and loaded in memory

Types of state :

Text and data - static state when the process first loads.

Heap - dynamically created during execution.

Stack - grows and shrinks, LIFO queue

What does a process look like?

The virtual address was created. It is used by the process.

The physical address locations in physical memory.

Address space == “in-memory representation of a process”

Page tables == “mapping of virtual to physical address”

Address space and memory management

Allocates the process in the memory where the memory has free space.

Process execution state

Program counter - stores the next process state to which process to go.

CPU registers

Stack pointer

Process control block(PCB) - contains the proces states / status details.

**Buffer overflow attack**

**Playlist 1 - Process control block**

It is created when process is created.

Certain fields are updated when process state changes - memory limits

Other fields change too frequently - process state, process number, program counter, registers, list of open files, priority, signal mask, CPU scheduling info..

**How is a PCB used ?**

Two process - P1, P2

P1 is running , P2 idle PCB\_P1 in CPU regs. Save PCB\_P1, Restore PCB\_P2

P2 starts running, P1 idle PCB\_P2 in CPU regs. Save PCB\_P2, Restore PCB\_P1

It helps in running multiple processes simultaneously.

**Context switch**

Switching the CPU from the context of one process to the context of another process.

They are expensive.

Direct costs - no. of cycles for load & store instructions

Indirect costs - COLD cache. Cache misses.

For better and complete the execution faster, LIMIT how frequently context switching is done.

**Process Lifecycle : States**

Processes can be running or idle.

The other states can a process be in -

new, ready, running, waiting, running, terminated.

**Process Lifecycle : Creation**

Mechanisms for process creation - fork, exec

Scheuler - schedules the processes and initializes the processes.

Fork - copies the parent PCB into new child PCB. child continues exec. at instruction after fork

Exec - replace child, load new program and start from first instruction.

**Role of the CPU scheduler -**

A CPU scheduler determines which one of the currently ready processes will be dispatched to the CPU to start running.

It must be EFFICIENT.

Preempt - interrupt and save current context

Schedule - run scheduler to chose next process

Dispatch - dispatch process and switch to its context

**Length of process -**

How long should a process run for? How frequently should we run the scheduler?

Useful CPU work = Total processing time / Total time

**I/O -**

I/O - process expecting the I/O request - to the I/O queue

Time slice expired

Fork a child - child executes

Wait for an interrupt - interrupt occurs

In these 4 - process goes again to ready state

**Inter process communication -**

Message passing IPC -

OS manages the processes - write (send), read (receive)

OS Overheads

Shared memory IPC -

Establishes a shared channel and maps it into each address space.

Processes directly read/write from this memory.

OS is out of the way.

**Playlist 2 - Threads and concurrency**

**Visual metaphor**

A thread is a active entity, executing unit of a process

runs simultaneously, many threads executing

Requires coordination, CPU, memory.

**Process and Threads -**

In a process, code, data, files, and regs. Stack in the same memory

Only one execution

While, In thread - contains more than one execution.

**Benefits of Multi-threading -**

Parallelization, and -> so, process speed up its execution, Reduce the time

Specialization -> hot cache - execution to each and every cache and all threads have the same memory.

Efficiency -> lower RAM requirement, cheaper IPC because all threads have the same memory.

**Benefits of Multi-threading : Single CPU -**

Two diffe. Threads t1, t2 on different memory addresses

If (t\_idle) > 2 \* (t\_cnxt-swch)

Then context switch to hide idling time

Two diffe. Threads t1, t2 on same memory addresses

T\_cnxt-switch threads < T\_cnxt-switch process

**Benefits of Multithreading : Apps and OS -**

In user - multi-threaded applications in web servers and databases

In kernel - multi-threaded OS kernel

Thread works on behalf of applications.

OS level services like daemons of drivers.

**Basic thread mechanisms -**

Thread data structure - identify threads, keep track of resource usage.

Mechanisms to create and manage threads

Mechanisms to safely coordinate among threads running concurrently in the same address space.

Mutual exclusion - exclusive access to only one thread at a time. Mutex.

Waiting on other threads - specific condition before proceeding, condition variable,

Waking up other threads from wait state.

**IO Management**

Operating system

* Has protocols
* Interfaces for device I/O
* Has dedicated handlers
* Device drivers, interrupt handlers
* Decouple I/O details from core processing
* abstract I/O device detail from applications

## **I/O Device Features**

* Control registers (accessed by CPU) - Command, Data Transfers, Status
* Microcontroller : device's CPU
* On device memory
* Other logic – example: analog to digital

## **Device drivers**

* per each device type
* responsible for device access management and control
* provided by device manufacturers per OS /version
* each OS standardizes interfaces - device independence, device diversity

## **Types of devices**

* Block example: disk
  + read/write blocks of data
  + direct access to arbitrary block
* Character example: keyboard
  + get/put character
* Network devices

OS representation of a device : special device file

UNIX like systems:

* /dev
* tmpfs
* devfs

Linux supports a number of pseudo "virtual" devices that provide special functionality to a system.