# Project:Shell

Task 1.5: Provide a concise and descriptive answer to the following questions.

Q1: What does the mmap() function do?

The linux manual will help with the basic definition of this function:

mmap, munmap - map or unmap files or devices into memory

The mmap() function creates a mapping in the virtual adress space of the process in which it is called.

The function takes the following parametres:

void \*mmap(void \*addr, size\_t length, int prot, int flags,

int fd, off\_t offset);

First of all, we see that this function is of type void – it doesn't return any parameter.

The addr argument specifies the starting adress where the new mapping will be created.

The length parameter contains the length of the new mapping – hence it must be greater than zero.

The prot argument can be used for selecting the desired memory protection of the mapping

It can have one or more of the following values / flags:

PROT\_EXEC: Pages may be executed.

PROT\_READ: Pages may be read.

PROT\_WRITE: Pages may be written.

PROT\_NONE: Pages may not be accessed.

The flags argument determines whether updates to the mapping are visible to other processes mapping the same region, and whether updates are carried through to the underlying file.

It can have one or more of the following values / flags:

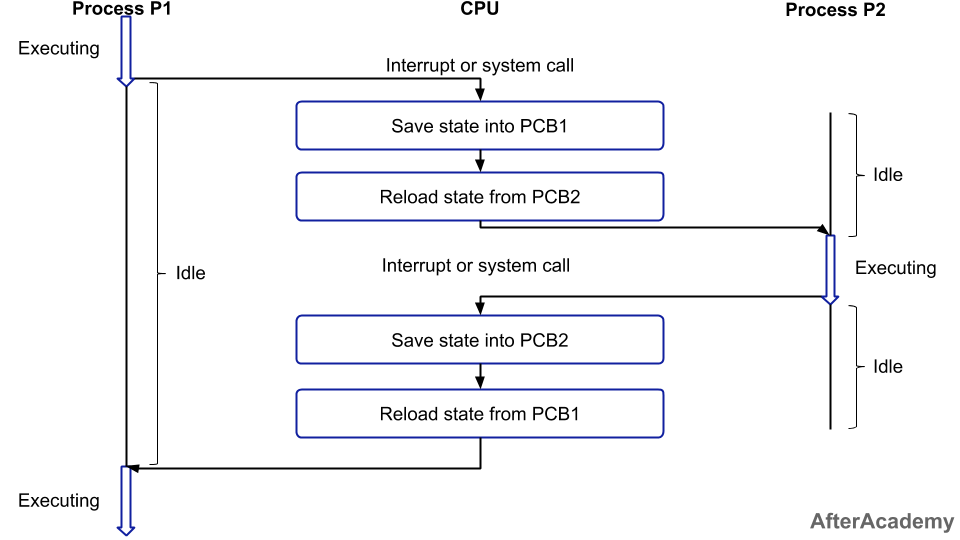
MAP\_SHARED, MAP\_SHARED\_VALIDATE,MAP\_PRIVATE, MAP\_32BIT, MAP\_ANON/ MAP\_ANONYMOUS

Q2: What happens during a context switch? Do we want the OS to perform many or few context switches? Explain.

A context switch is a process that involves switching of the CPU from one process or task to another. In this phenomenon, the execution of the process that is present in the running state is suspended by the kernel and another process that is present in the ready state is executed by the CPU.

A context is the contents of a CPU's registers and program counter at any point in time. Context switching can happen due to the following reasons:

* When a process of high priority comes in the ready state. In this case, the execution of the running process should be stopped and the higher priority process should be given the CPU for execution.
* When an interruption occurs then the process in the running state should be stopped and the CPU should handle the interrupt before doing something else.
* When a transition between the user mode and kernel mode is required then you have to perform the context switching.



Context switching is crucial for multitasking / multiprogramming – it gives the illusion that more than one process is executed at the same time.

However, as context switching takes some amount of time to execute – it can be costly in terms of processor time. That is the the time required to save the context of the process that is in the running state, and then getting the context of another process. During this time, the CPU does no useful work – and in this way, context switching represents a pure overhead.

These are the reason why we want to be as compact as possible – and doing much with less context switches.