National Institute of Technology, Raipur

Department of Computer Science and Engineering

Operating Systems-Term Paper

COURSE- B.Tech. **SEMESTER-** 5th

NAME- SHASHWAT AWASTHI

ROLL NO. 18115072

FACULTY IN-CHARGE- Dr. DILIP SINGH SISODIA

PROJECT TOPIC- I/O Interlock and Page Locking, Pinning and Need of Page Locking.

Introduction-

Virtual Memory is a technique of using more memory that is actually present in a memory. Programs which are loaded into the Main memory can be sometimes larger then the main memory, thus it will be a problem storing it in the main memory. Thus, Virtual memory provides us a memory management technique in which we can access the program easily. The contents of the memory of the program not currently in use can be transferred to the disk memory of the system and can be fetched for use from there whenever they are required. This temporary of process of memory management creates a virtual visualization of infinite memory in program execution and thus is a very useful method.

Demand Paging is a part of Virtual Memory management process. According to what Virtual memory concepts say, only that amount of memory or pages need to be in memory which are currently used by the program for its execution. Rest all the pages and memory can be stored in secondary memory or Hard disk of the computer and may be called in whenever necessary. So, for this, all pages are initially loaded in the secondary memory for execution of the program. Whenever the program execution starts, the pages are referred from the secondary memory in accordance with the usage of the program. Whenever the first time its referred, it is called **Pure Demand Paging**. Likewise, when more pages are referred, it is called Demand Paging. The name demand paging is called so, as pages are only loaded in primary memory, whenever the program "demands" them.

Whenever the demand paging is used, the page being referred to by the system is been supplied by the secondary memory, without keeping a record whether some other process might be needing it to for further execution. This might lead to error situation as if a page been required by some other process like Input or Output is already been loaded into the memory, then that process might fail and might hamper the working of the computer `indefinitely.

Thus, it is required to have some pages in the secondary memory locked, so that whenever they are required by some other process, can be referred to them first. This can be achieved by page locking, meaning locking some parts of pages in computer memory which are to be used for some high priority process. This assignment is focussed on those techniques, which are used to get out of these situations in your system, and also discussing about the problems if Locking is not done.

I/O interlock and Page Locking-

I/O Interlocking

Demand paging always provides pages to the primary memory whenever they are being referred from the secondary memory. This leads to some pages being not available in the memory whenever a high priority process like input/output calls for it. Thus, it is beneficial if we lock some pages into the memory of the system, so that they cannot be part of demand paging whenever a page fault occurs and a page replacement algorithm is called.

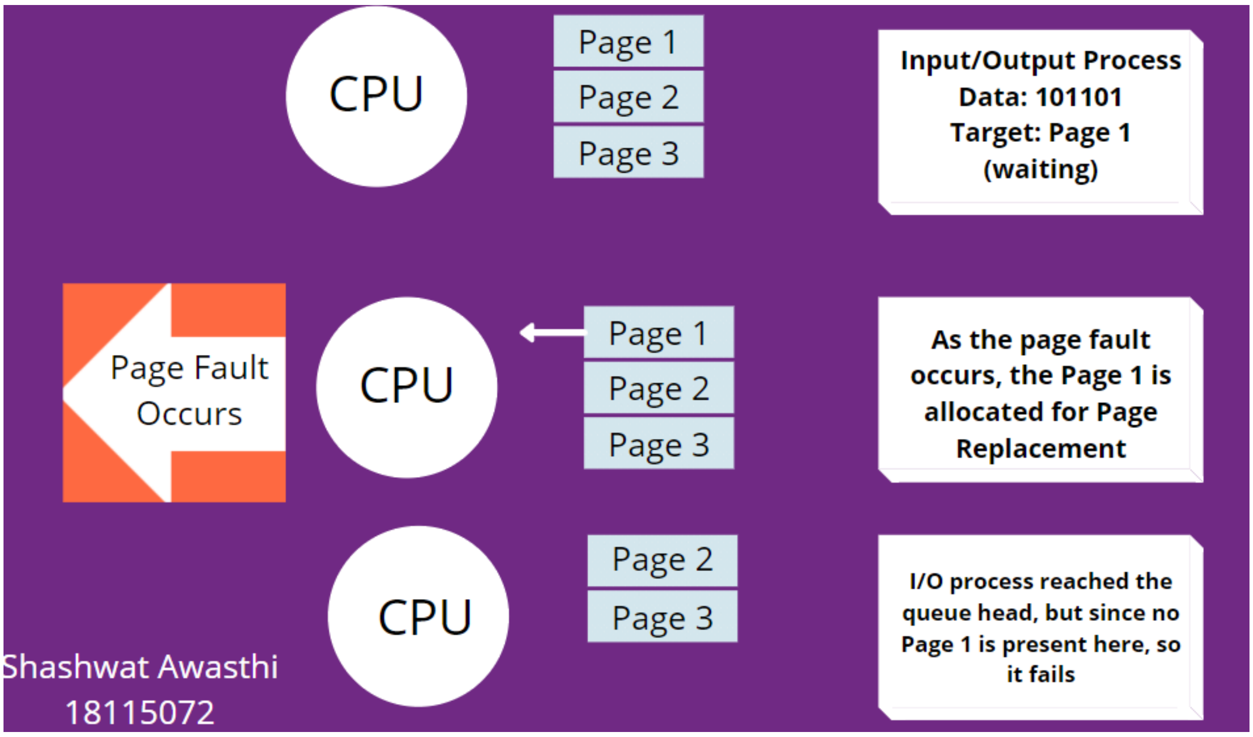
Lets take an illustration to prove what can happen if this is not present. An I/O processor is responsible for the execution of I/O between the secondary memory and Input /Output devices. So the sequence of events can be-

- When an Input/ Output process is executed, the process gets the data to be transferred to the system and also the address of the page or memory from which the data has to be taken from or loaded into.
- The I/O process issues an interrupt request to the CPU(Interrupts are message signal sent to the CPU whenever any process needs to do something), and the CPU places that interrupt request in a queue of various processes to be executed. The process waits for it’s turn to be executed
- Meanwhile, the CPU is also running other processes in the system. Suppose, a page fault occurs in a process, and the CPU is called for pages by the page replacement algorithm
- The CPU provides the pages containing information which is to be utilised by the I/O process which is waiting for its turn in the queue.
- So, that page is “paged out” and is been given to be utilised by some other process whose page fault has occurred by the CPU. This means, now that page is in the main memory and is being used by the current program in execution.
- After some time, when the turn of the I/O process, which was waiting in the queue, comes, then the I/O process requests for the page where it was required to take the data from or load the data.
- Since the CPU has already allocated that page to a process earlier, this means that page might either be used by that process currently or the information contained in that page has been overwritten.

- Thus, the I/O process will fail in its execution due to the fact that the page which was required by the I/O interrupt was not available.

This sequence of events is referred to as the I/O interlocking. Due to this, the I/O interrupt fails to perform its process and thus the process depending on it also fails.

The diagram below illustrates this Interlock-


CPU
Page 1
Page 2
Page 3
Input/Output Process
Data: 101101
Target: Page 1
(waiting)
Page Fault Occurs
CPU
Page 1
Page 2
Page 3
As the page fault occurs, the Page 1 is allocated for Page Replacement
CPU
Page 2
Page 3
I/O process reached the queue head, but since no Page 1 is present here, so it fails
Shashwat Awasthi
18115072

Page Locking

We need various solutions to get rid of the above problem. To do that, we basically can use two methods-

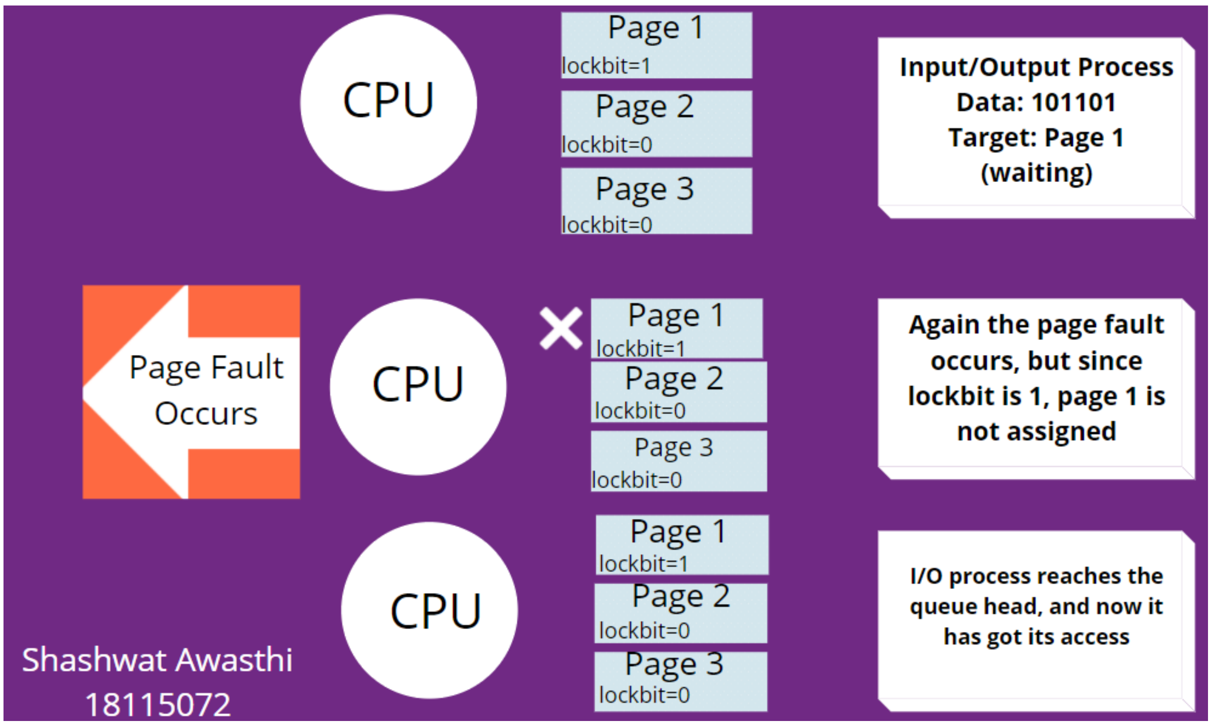
1. Since the I/O process takes place between system memory and the I/O devices, we can copy data between the system memory and the memory utilised by the user. Then the input output process can work separately.
2. We can use the Page Locking system. A lock bit can be assigned to each page and thus we can lock the page in the user memory which is to be utilised by the I/O process. If we lock the page, the CPU cannot select if for replacement when the page fault occurs.

Thus, here we use the Page Lock mechanism to get rid of the Interlocking problem. The steps involved in page locking are-

- Every page stored in the memory has a specific lock bit attached to it which is set to 0 initially, that means the page is not locked. If the lock bit for page is 1, it means that the page is locked.
- When a process of high priority enters the queue, the lock bit for the specific page which needs to be targeted is set to 1, that means the page gets locked.
- This locked page gets reserved for that specific process, and thus when the process is ready for execution, it uses the page it needs.
- After the process execution completes, then the lock bit is set to 0, that means the page is unlocked and can be used by other processes as well. In this way, we don't need to have more memory for separate copies of memory.

Thus, since we set the page lock bit to 0 whenever the process gets completed, so we save more memory and also can no Interlocking occurs. Thus efficiency can be maintained of the system.

Below is an illustration of page locking-

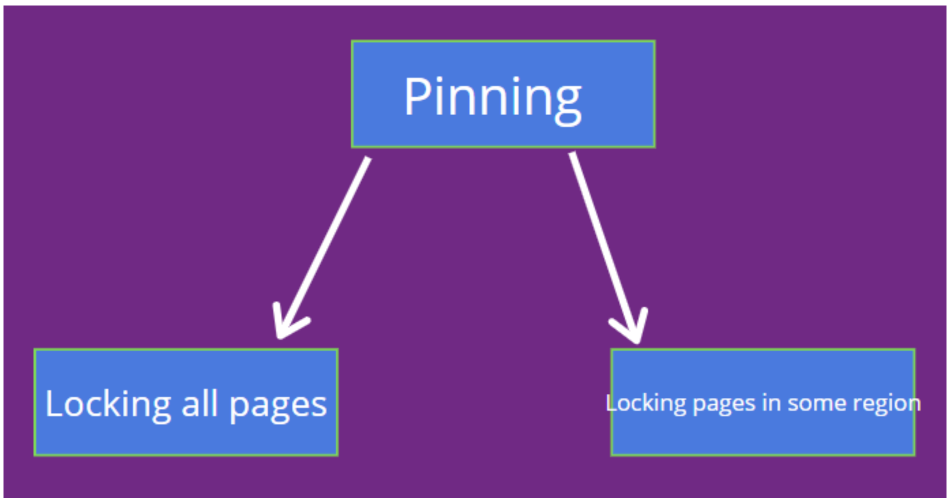


Pinning-

The above concept of lock bit can be extended to various other situations as well. It is not only restricted to locking while working with a I/O interlocking mechanism. The various situations where a page locking scenario can be used are-

- The User application program Processes might need to lock pages in the memory for their use. It is not restricted only to I/O processes. E.g. While a game is being played, all pages having data related to the game might be locked as it might be needed immediately.
- Database processes in the system might use it to lock pages which have data related to all the available process in the memory as well as switching of data between primary and secondary memory.
- Since the Kernel is the core of the operating systems, all the pages containing data related to kernel, might be locked, to ensure proper running of the Operating system. Page fault related to kernel can cause problems, thus it is needed.

This process of locking the pages in the main memory and allow them to be explicitly exempted from the page replacement process during paging, is called **Pinning. Pinning guarantees Immediate availability of pages to a process when the CPU is interrupted or referenced for.** Process being executed in real-time scenarios use this concept as they might require some amount of data immediately from the system, thus the pages are specifically pinned for that process, to be specifically used. If pinning is not down and a process requires some pages instantly, some of the time which could have been utilised for the execution of the process maybe spent out in retrieval of pages.



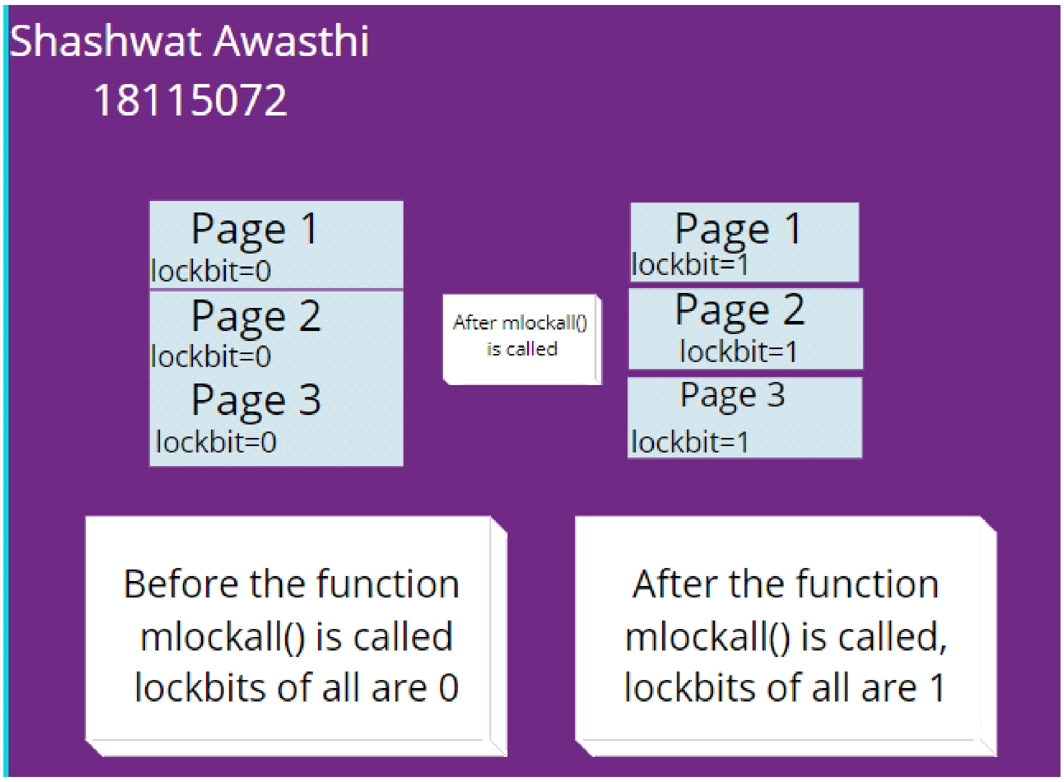
Pinning can be categorised into two major ways-

- **Locking all the pages**- Whenever a process enters the queue to get processed, we can lock all the pages which are to be demanded by the process for its execution. Any library/ data which is being used in shared mode is also completely locked for the process.

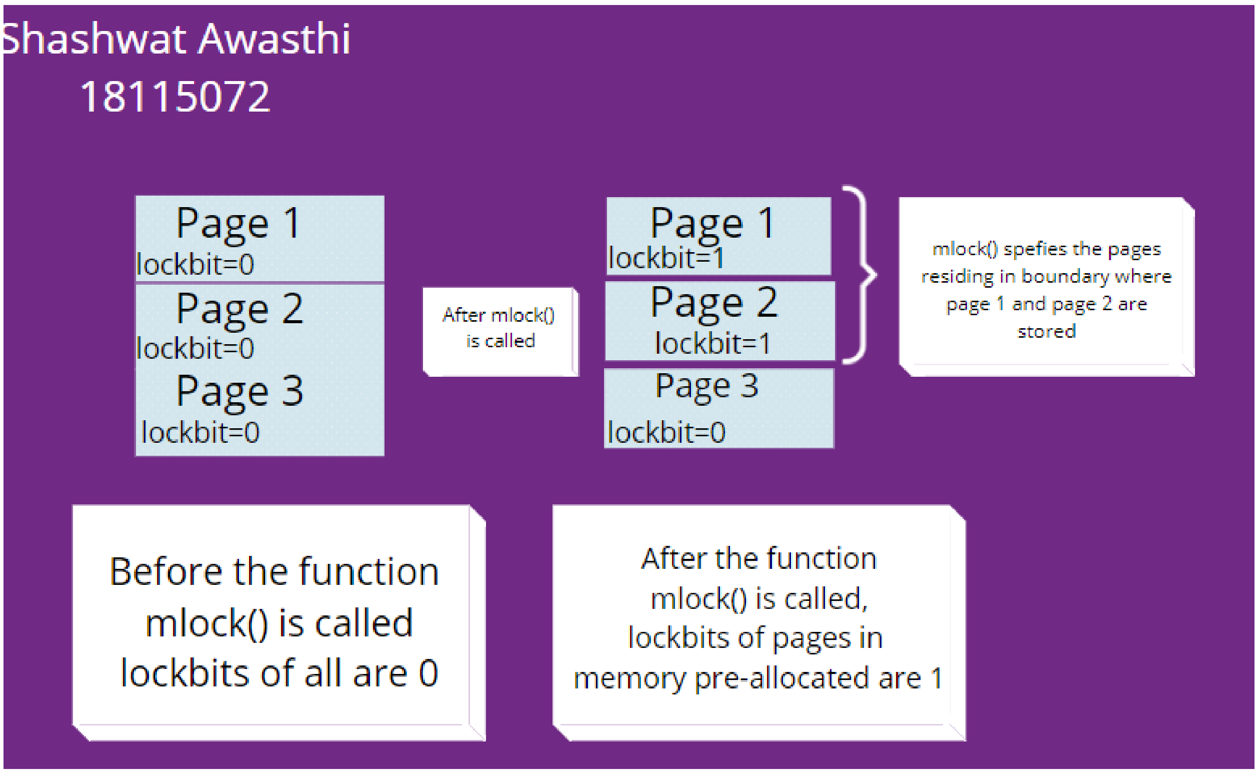
  Locking of all pages in memory can be achieved by a function **mlockall().** This function accepts the flags for which the pages needs to be pinned in the memory. e.g. Linux Operating system has two kinds of flags-

  1. MCL_CURRENT flag: This flag guarantees that all the current pages stored in the memory in response to the incoming process might be pinned in the memory.
  2. MCL_FUTURE flag: This flag guarantees that the pages which are allocated in memory in future after the function is called are pinned to memory. With this flag, the memory pages created after the call of function are also locked for the process.

Thus, the mlockall() function takes these two flags and pin the pages related to this in memory.


Shashwat Awasthi
18115072
Page 1
lockbit=0
Page 2
lockbit=0
Page 3
lockbit=0
After mlockall() is called
Page 1
lockbit=1
Page 2
lockbit=1
Page 3
lockbit=1
Before the function mlockall() is called lockbits of all are 0
After the function mlockall() is called, lockbits of all are 1

- **Locking pages in some specific memory region-** In this, we lock only pages in a specified memory region of the secondary memory. We choose a specific memory region, the pages belonging to which when requested by the process will be locked.
  This can be achieved by calling the function **mlock().** The function mlock() accepts arguments which specify the pre-allocated region where the pages need to be locked and on calling of the function, only the pages residing in that particular region are locked.


Shashwat Awasthi
18115072
Page 1
lockbit=0
Page 2
lockbit=0
Page 3
lockbit=0
After mlock() is called
Page 1
lockbit=1
Page 2
lockbit=1
Page 3
lockbit=0
mlock() spefies the pages residing in boundary where page 1 and page 2 are stored
Before the function mlock() is called lockbits of all are 0
After the function mlock() is called, lockbits of pages in memory pre-allocated are 1

Thus, **Pinning** guarantees that pages which are requested by a particular process might be locked/pinned in the memory for their use and shall be exempted from page replacement whenever dynamic paging happens.

<u>Need for page locking-</u>

Page locking is quite important for processes as it guarantees an instant request approval for a process whenever they reference for a page using an interrupt, thus saving valuable amount of time. In a nutshell, page locking help us improved the performance of our system.

Various advantages of Page locking are-

- Helps in removing the I/O interlocking and thus saving time for execution of the process.
- Normal Page replacement: Sometimes when a low-priority process enters the ready queue and asks for a page from secondary memory, the CPU assigns it to that process and when the high priority process executing currently faults and asks for a new page, the CPU cannot provide it. Thus, page locking can help.
- Page locking help reduce time overhead, and thus help in reducing the completion time of a process.
- Page locking will be very beneficial when used by the kernel of Operating system, as the kernel is most important part and thus its requirement should be fulfilled immediately.

Drawbacks of page locking-

- If the system is not able to change the status of the lock bit, in that case the page might be locked forever and thus will become useless.
- If two process are simultaneously asking for locking on a page, it is difficult to decide the request of which process is to be fulfilled.

Conclusion-

Thus, we can conclude that page locking Is a very important task which helps in improving upon the system’s overall performance and also ensures a synchronized and error free execution of processes.