CASE STUDY

OPERATING SYSTEMS(CSE 316)



Student Name : Somepalli NagaBrahma Chaitanya

Student Id : 11810677

Email : [somepallichaitanya6@gmail.com](mailto:somepallichaitanya6@gmail.com)

Github Link :

**CONTENT**

* **Introduction**
* **Description**
* **Algorithm**
* **Code Snippet**
* **Complexity**
* **Test cases**
* **Github Repository**
* **Conclusion**

**INTRODUCTION :**

In [computer](https://en.m.wikipedia.org/wiki/Computer) [operating systems](https://en.m.wikipedia.org/wiki/Operating_systems), **demand paging** (as opposed to [anticipatory](https://en.m.wikipedia.org/wiki/Paging#Page_replacement_techniques) paging) is a method of [virtual memory](https://en.m.wikipedia.org/wiki/Virtual_memory) management. In a system that uses demand paging, the operating system copies a disk [page](https://en.m.wikipedia.org/wiki/Paging) into physical memory only if an attempt is made to access it and that page is not already in memory (*i.e.*, if a [page fault](https://en.m.wikipedia.org/wiki/Page_fault) occurs). It follows that a [process](https://en.m.wikipedia.org/wiki/Process_(computing)) begins execution with none of its pages in physical memory, and many page faults will occur until most of a process's [working set](https://en.m.wikipedia.org/wiki/Working_set) of pages are located in physical memory.

**DESCRIPTION :**

Demand paging follows that pages should only be brought into memory if the executing process demands them. This is often referred to as [lazy evaluation](https://en.m.wikipedia.org/wiki/Lazy_evaluation) as only those pages demanded by the process are swapped from [secondary storage](https://en.m.wikipedia.org/wiki/Secondary_storage) to [main memory](https://en.m.wikipedia.org/wiki/Main_memory). Contrast this to pure swapping, where all memory for a process is swapped from secondary storage to main memory during the process startup.

**Advantages:**

Demand paging, as opposed to loading all pages immediately:

1. Only loads pages that are demanded by the executing process.
2. As there is more space in main memory, more processes can be loaded, reducing the [context switching](https://en.m.wikipedia.org/wiki/Context_switch) time, which utilizes large amounts of resources.
3. Less loading latency occurs at program startup, as less information is accessed from secondary storage and less information is brought into main memory.
4. As main memory is expensive compared to secondary memory, this technique helps significantly reduce the bill of material (BOM) cost in smart phones for example. Symbian OS had this feature.

**Disadvantages:**

* Individual programs face extra latency when they access a page for the first time.
* Low-cost, low-power [embedded systems](https://en.m.wikipedia.org/wiki/Embedded_system) may not have a [memory management unit](https://en.m.wikipedia.org/wiki/Memory_management_unit) that supports page replacement.
* Memory management with [page replacement algorithms](https://en.m.wikipedia.org/wiki/Page_replacement_algorithms) becomes slightly more complex.
* Possible security risks, including vulnerability to [timing attacks](https://en.m.wikipedia.org/wiki/Timing_attack); see Percival, Colin (2005-05-13). [*"Cache missing for fun and profit"*](http://css.csail.mit.edu/6.858/2014/readings/ht-cache.pdf) *(PDF)*. BSDCan 2005. (specifically the virtual memory attack in section 2).
* [Thrashing](https://en.m.wikipedia.org/wiki/Thrashing_(computer_science)) which may occur due to repeated page faults.

**ALGORITHM :**

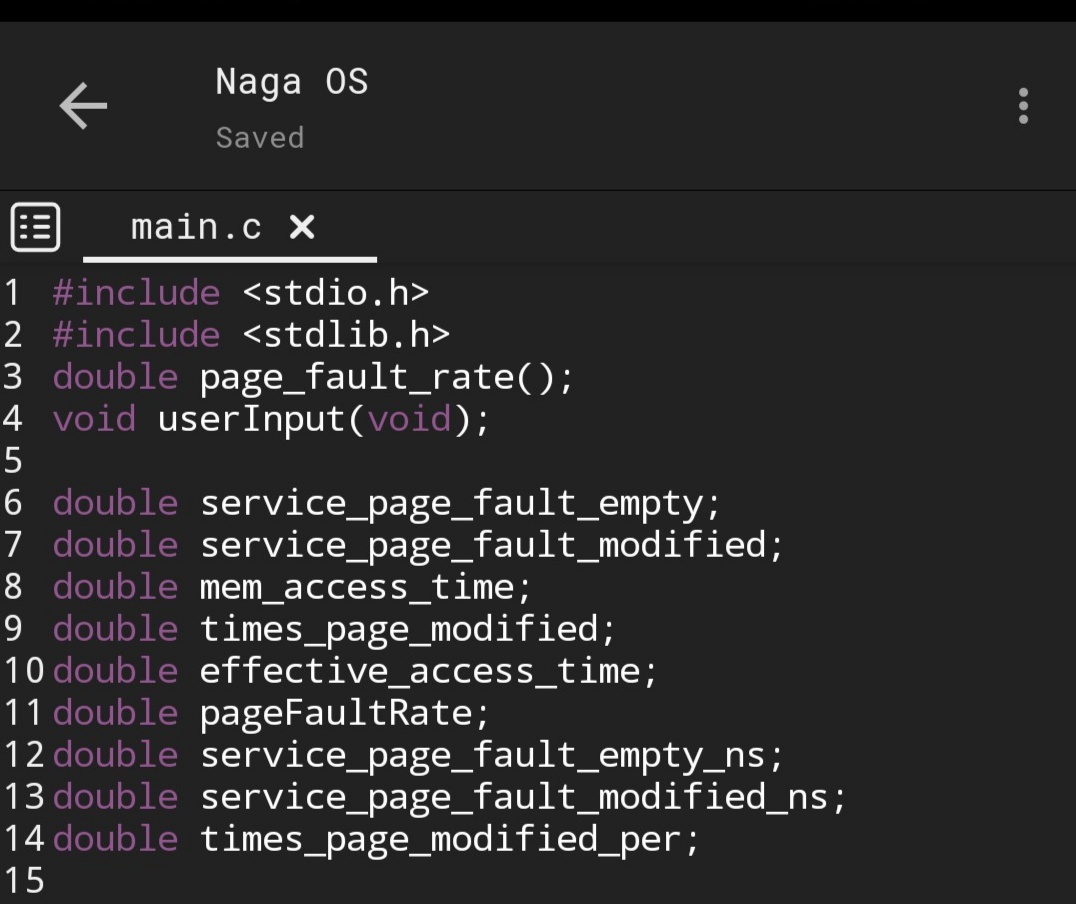
* Attempt to access page.
* If page is valid (in memory) then continue processing instruction as normal.
* If page is invalid then a **page-fault trap** occurs.
* Check if the memory reference is a valid reference to a location on secondary memory. If not, the process is terminated (**illegal memory access**). Otherwise, we have to **page in** the required page.
* Schedule disk operation to read the desired page into main memory.
* Restart the instruction that was interrupted by the operating system trap.

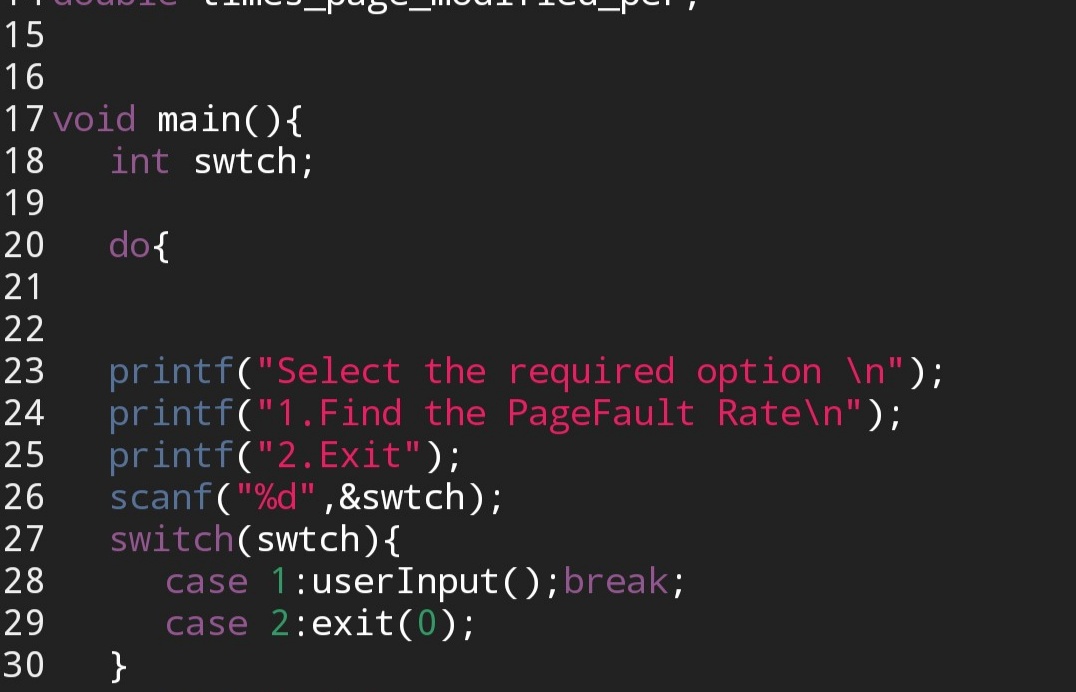
**COMPLEXITY :**

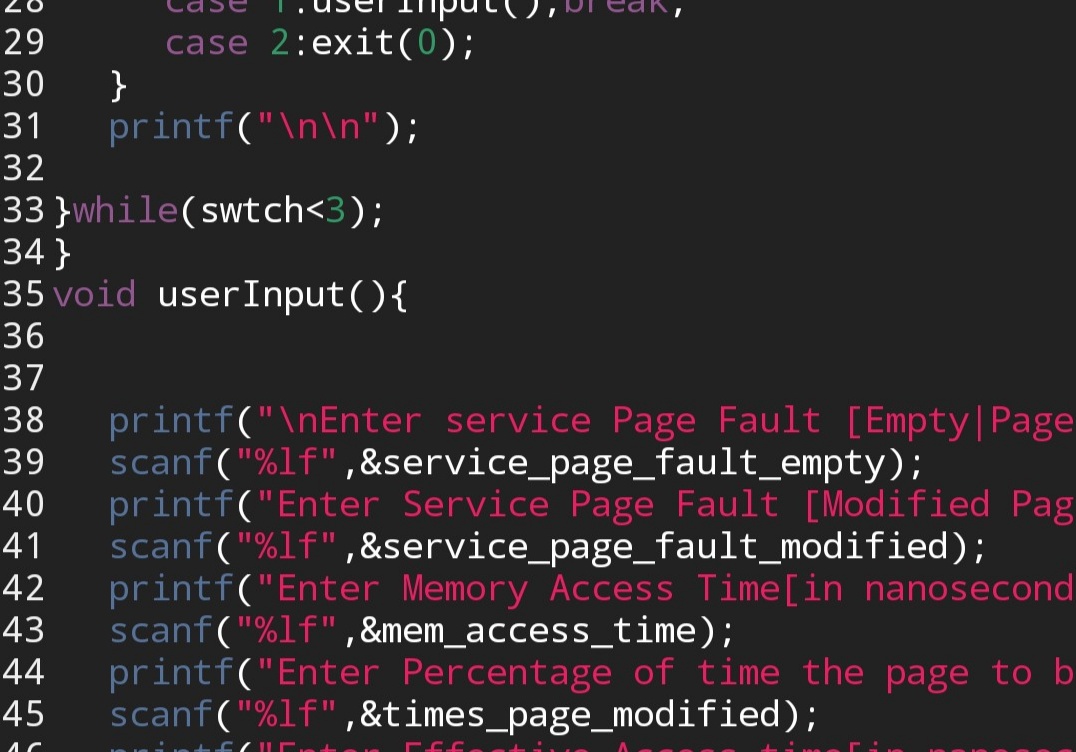
 In the demand paged memory allocation one programmer assumes  infinite  memory  called virtual memory. In which the  Job  operates  under demand.

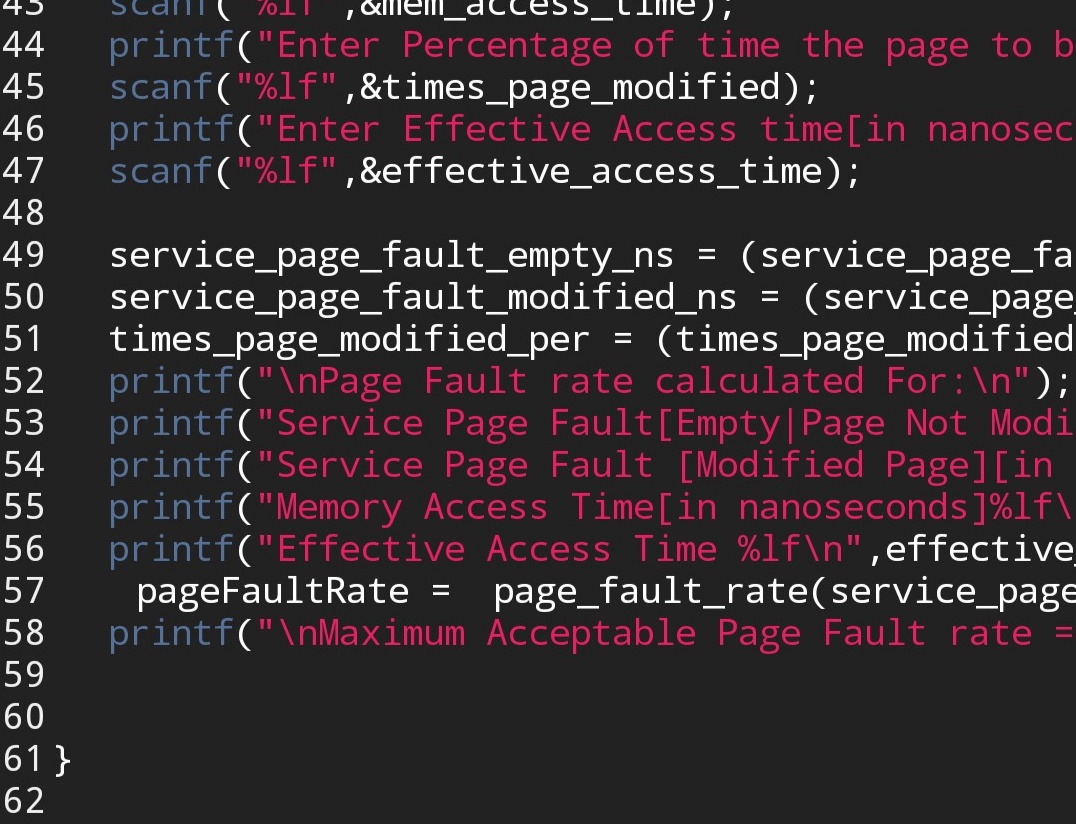
 Paging  virtual memory is large in size then the  available  memory the demand paging allocation is shown in figure.

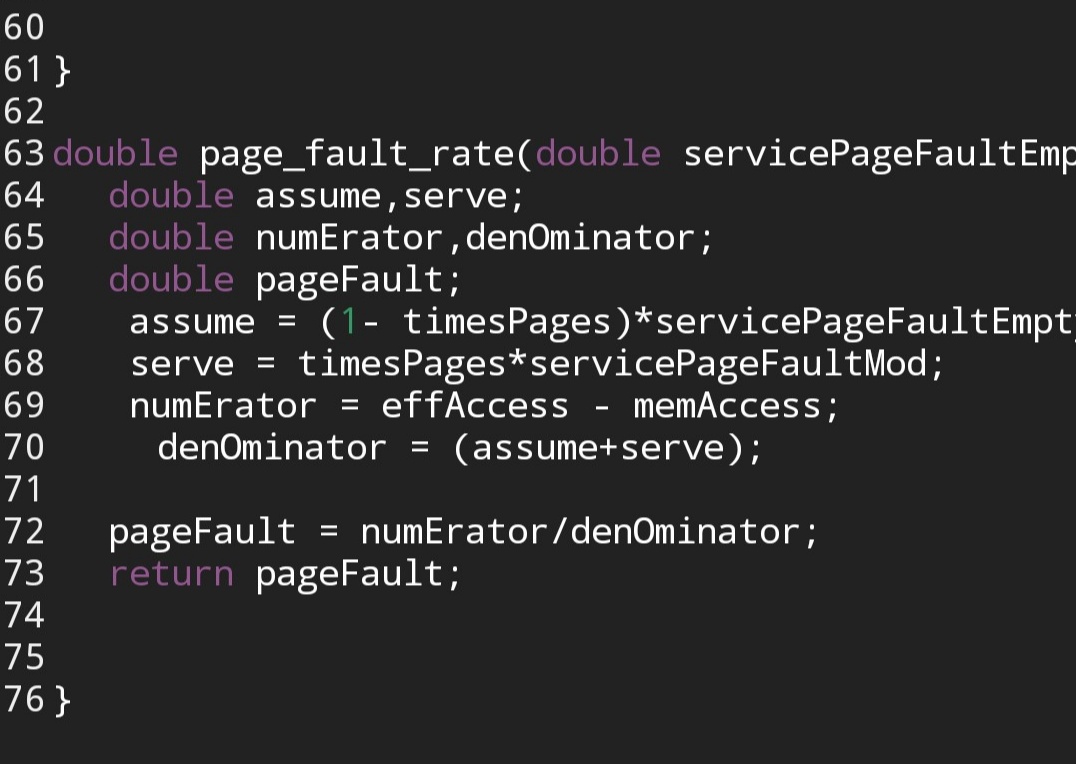
In demand pages memory allocation a particular page is swapped from the secondary storage device if it is required for execution.

**CODE SNIPPET:**

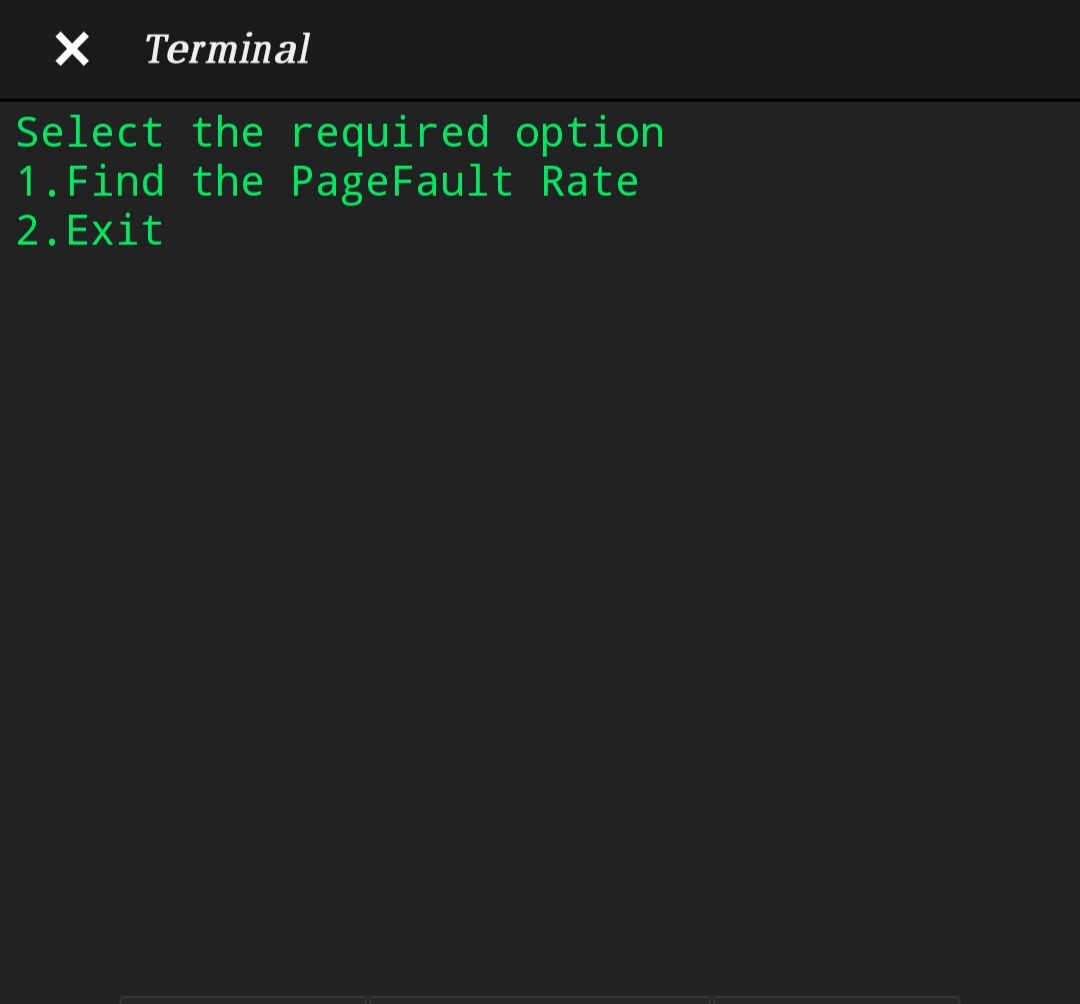


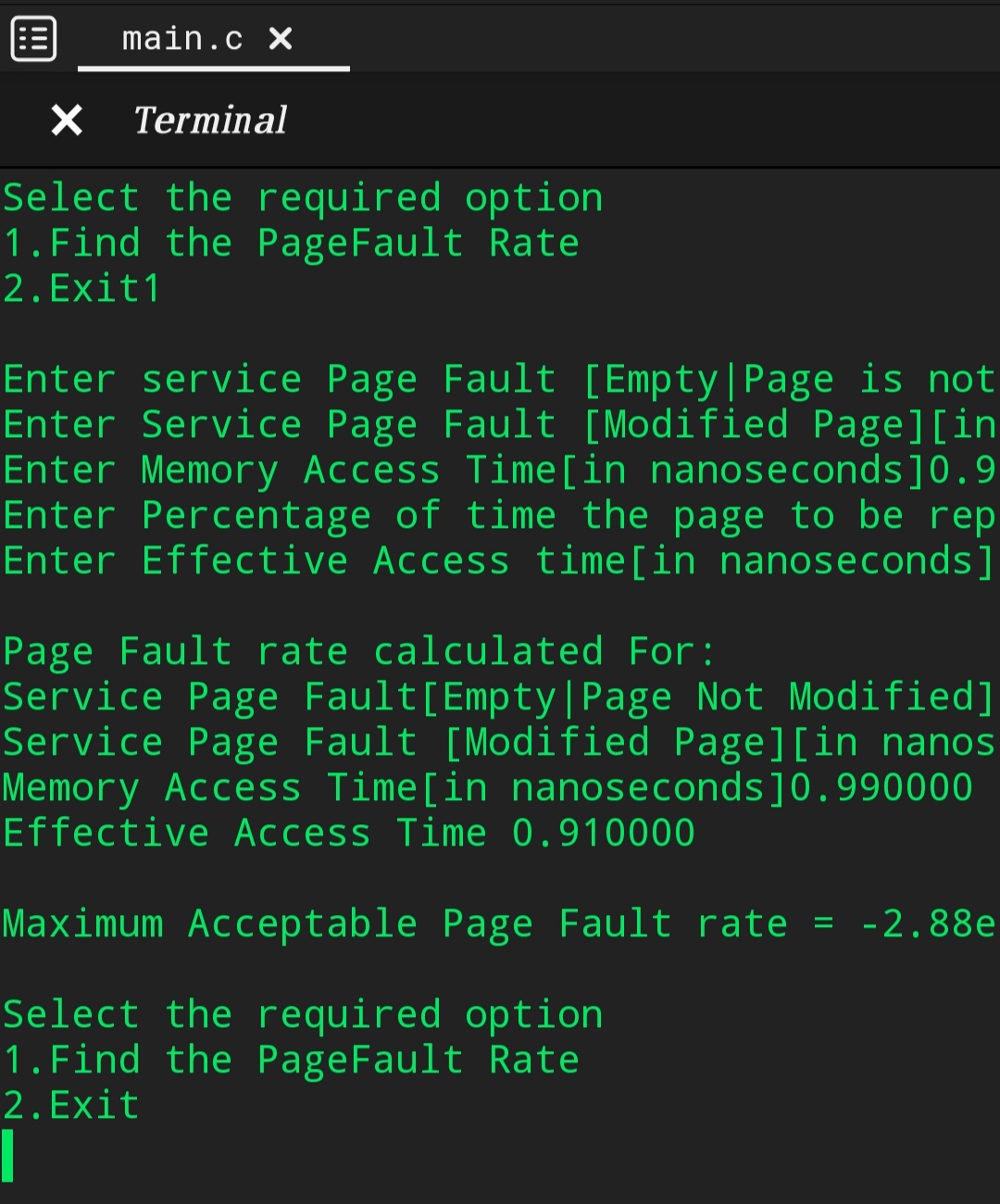




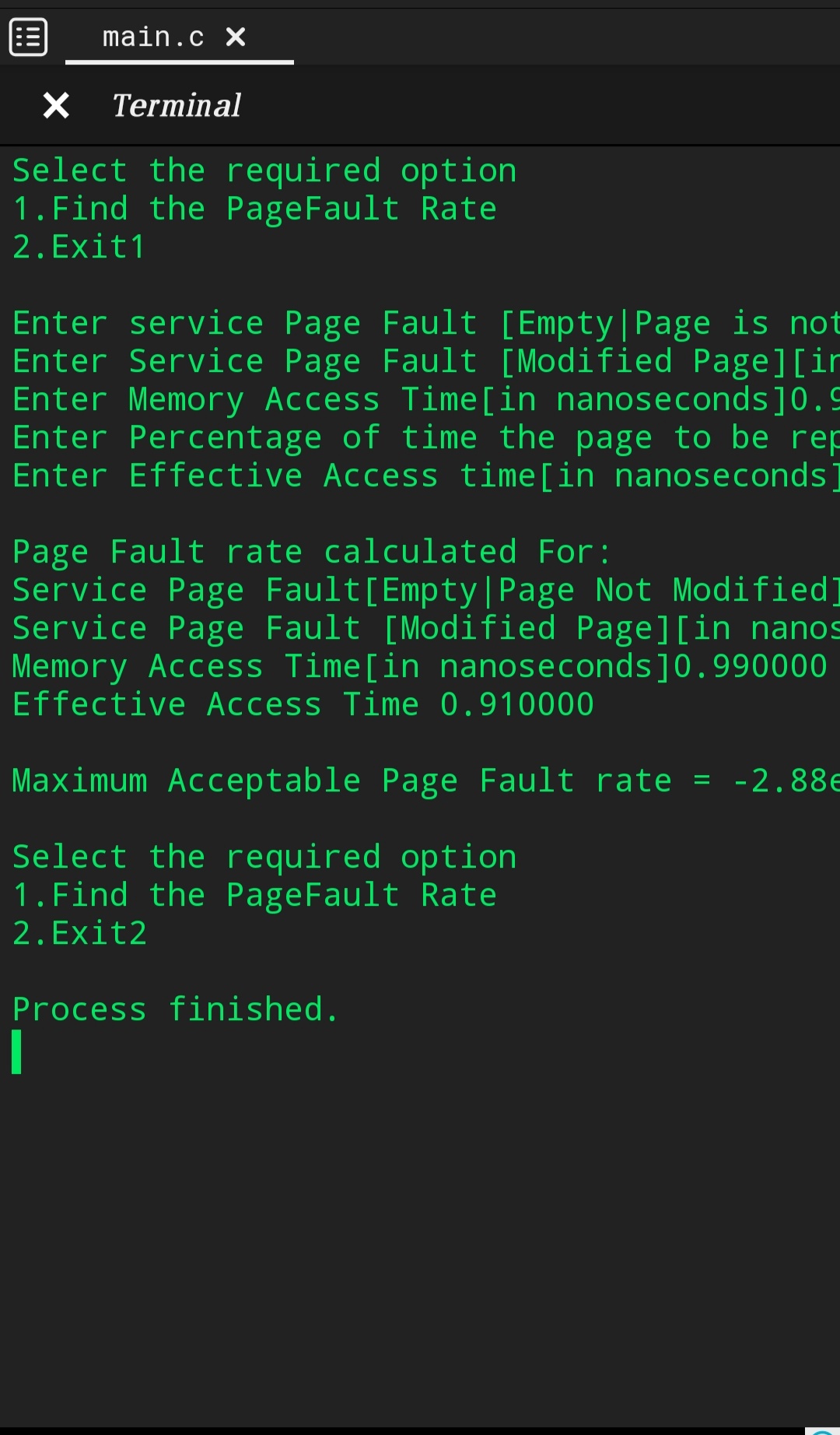


**TEST CASES :**

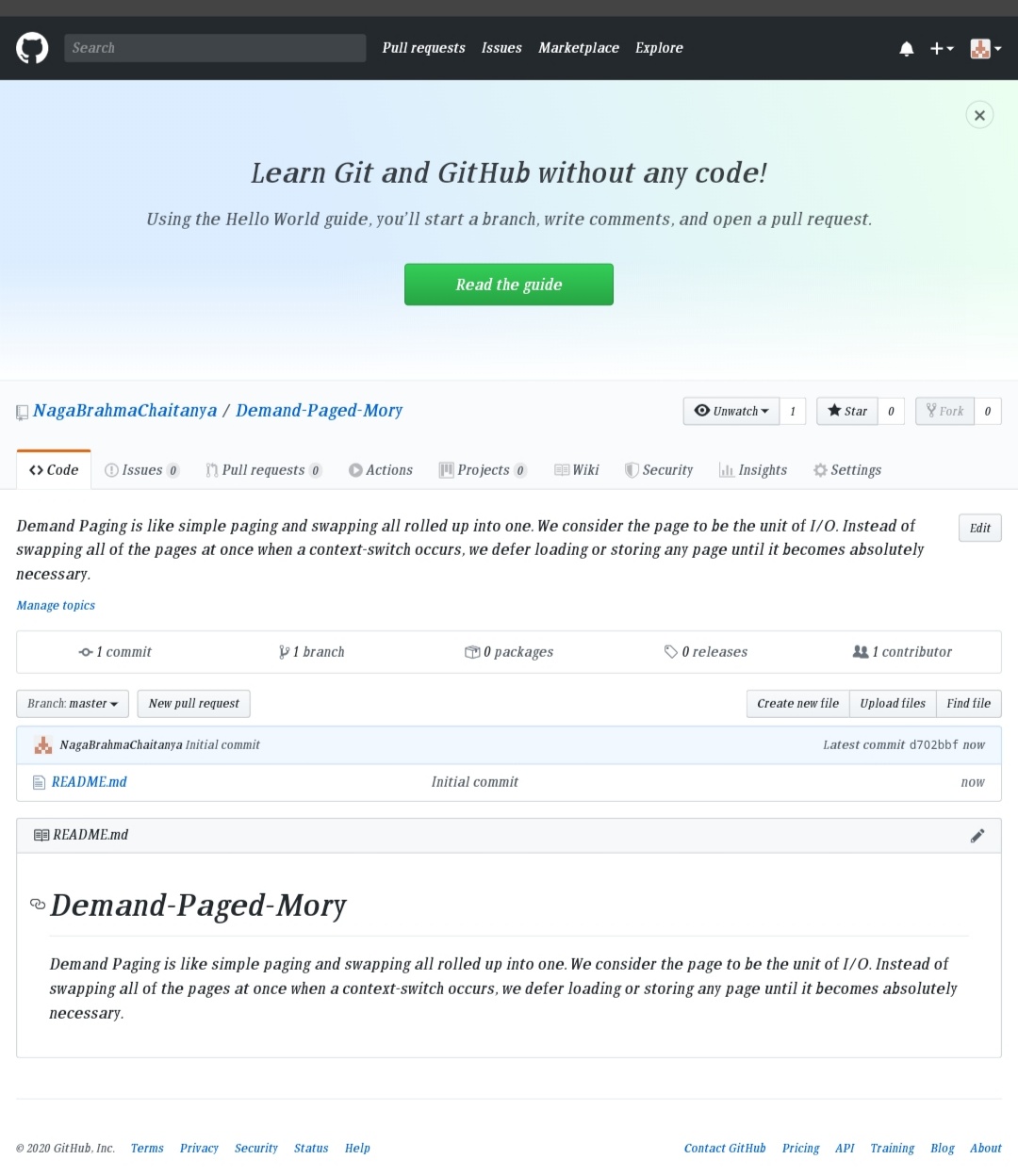
If we run the program we will get the following as available options:

After the available options we have to select if we want to update or do something by selecting option 1

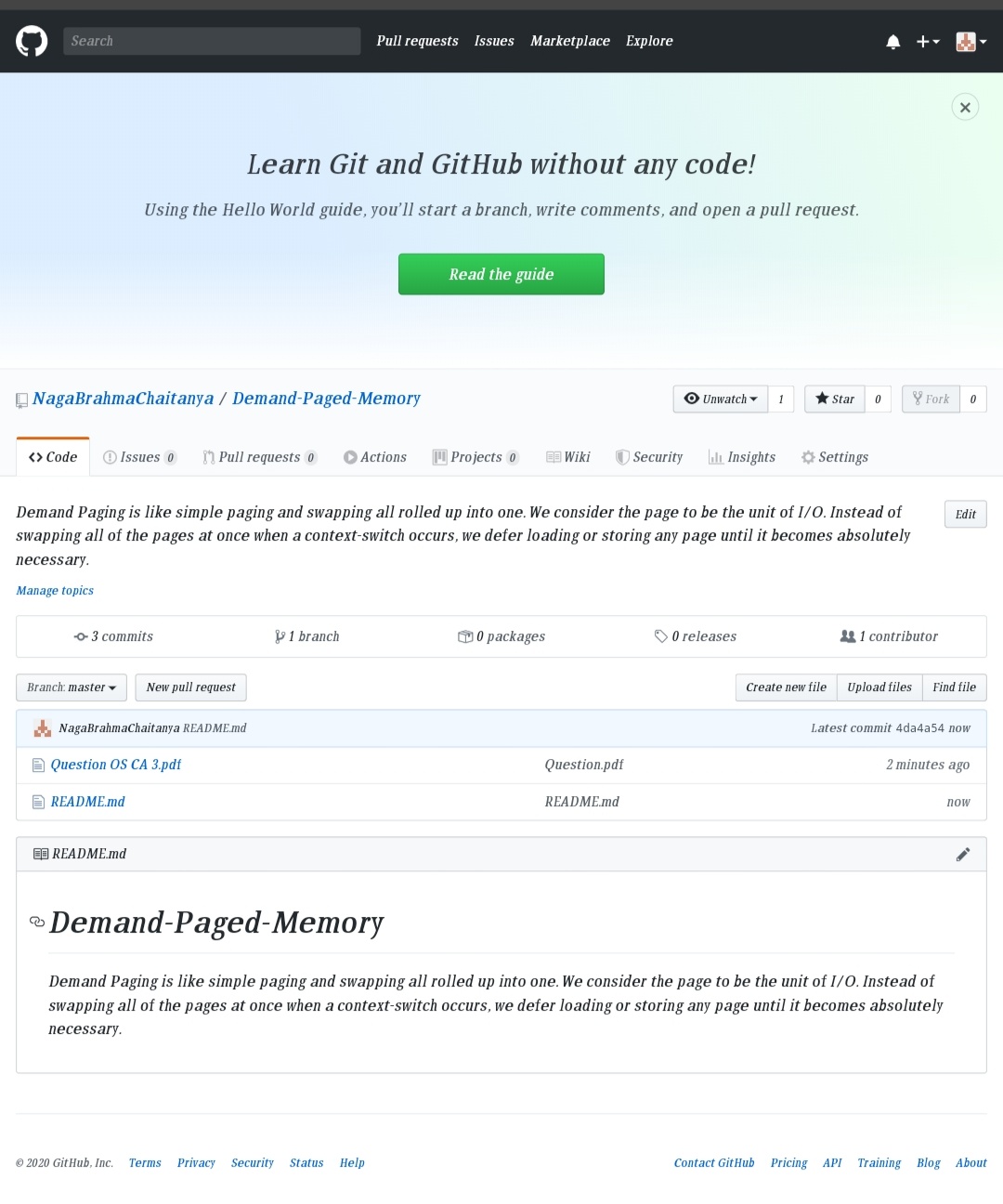
Then the process will be finished after it was the page fault rate was calculated:



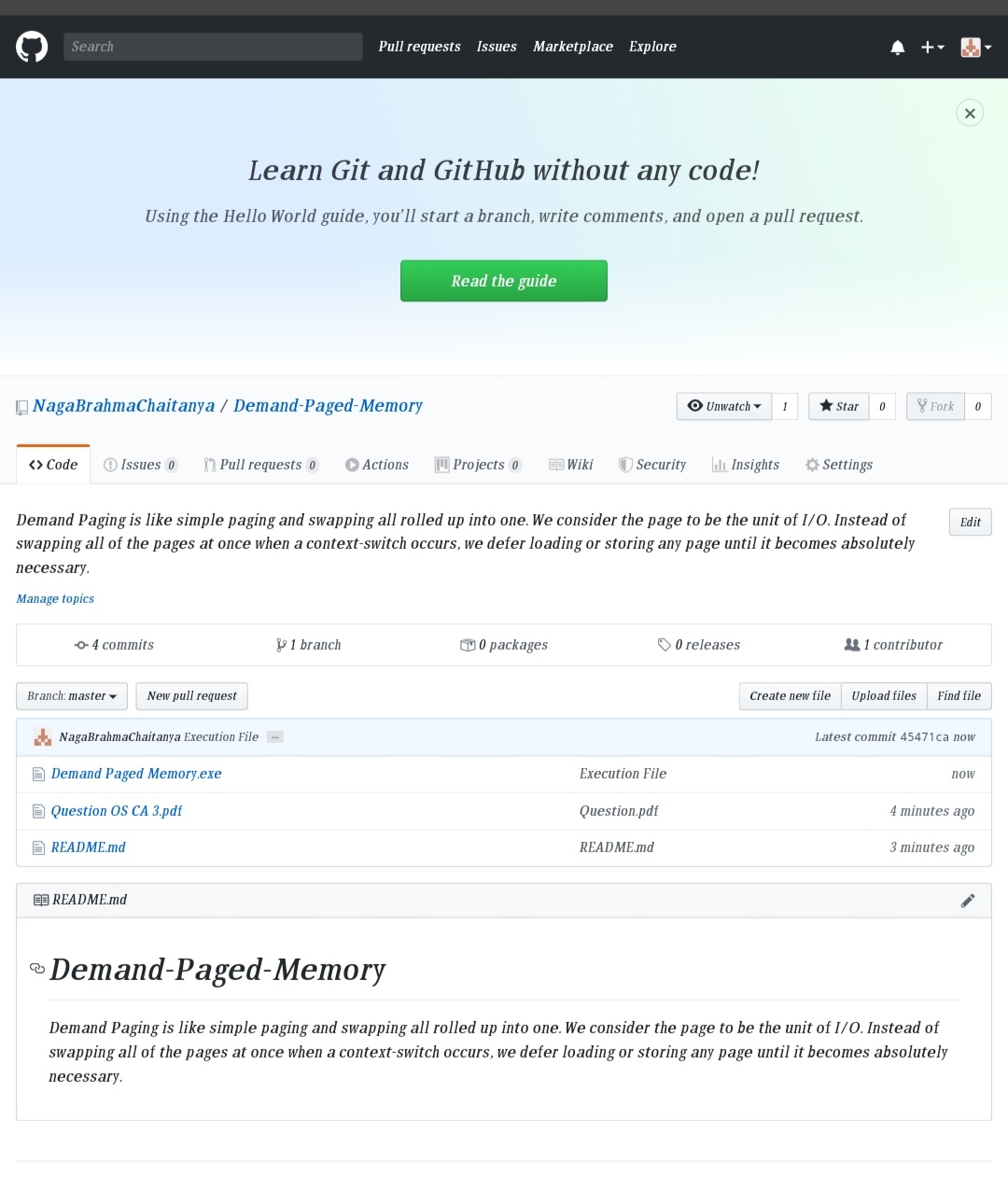
**GITHUB REPOSITORY:**

**First Commit:**

**Second Commit:**

****

**Third Commit:**

****

**CONCLUSION :**

Commonly, to achieve this process a [page table](https://en.m.wikipedia.org/wiki/Page_table) implementation is used. The page table maps [logical memory](https://en.m.wikipedia.org/wiki/Logical_memory) to [physical memory](https://en.m.wikipedia.org/wiki/Physical_memory). The page table uses a [bitwise](https://en.m.wikipedia.org/wiki/Bitwise_operation) operator to mark if a page is valid or invalid. A valid page is one that currently resides in main memory. An invalid page is one that currently resides in secondary memory.