CoE 135 (Operating Systems): Problem Set 2

Salmon, Paulino III I.

2015-11557

paulino.salmon@eee.upd.edu.ph

I. HONOR CODE

I pledge that I answered all the items in this Problem Set to the best of my understanding, and with the aid of properly cited resources and collaborators in problems I could not solve on my own. I made sure that I cited all resources and stated the names of all those who guided me in answering all the items along with their invaluable contributions.

- Paulino I. Salmon III, 2015-11557

II. ADDRESS TRANSLATION

- *A.* (10 pts) Consider the following Segment Table:

Segment Number	Base	Length
0	220	500
1	2000	15
2	100	90
3	1333	555
4	1969	71

What are the physical addresses of the following logical addresses given the segment table above?

- Segment 0, Address 430
  430 < 500 (Valid), 430 + 220 = **650**
- Segment 1, Address 10
  10 < 15 (Valid), 10 + 2000 = **2010**
- Segment 2, Address 500
  500 !< 90 **(Invalid)**
- Segment 3, Address 400
  400 < 555 (Valid), 400 + 1333 = **1733**
- Segment 4, Address 96
  96 !< 71 **(Invalid)**

  Self Grade For #: 4

- *B.* (5 pts) A computing machine provides its users with a virtual-memory space of 2^{32} bytes. The computer has 2^{18} bytes of physical memory. The virtual memory is implemented by a paging scheme, and the page size is 4096 bytes. A user process requests access to the virtual address 11123456. Determine and explain how the system establishes the corresponding physical address.
  - Page sizes are typically represented in powers of two for the reason that it's easier to divide the binary format of a number to represent both the page number and the page offset. If it is given that the page size is $log_2(4096) = 2^{12}$, the remaining bits will therefore be given to the page table, $32 - 12 = 20; 2^{20}$.

  Self Grade For #: 3

- *C.* (5 pts) You are given a demand-paged memory management scheme. It takes 10ms to service a page fault if an empty page is available or the replaced page is not modified, and it takes 18ms if the replaced page is modified. Memory access time takes 80ns. If the page to be replaced is modified 70% of the time, what is the maximum acceptable page-fault rate for an effective access time of 250ns?
  - Let *n* be equal to the page-fault rate (miss ratio)
    250ns = (*n*)[(10ms)(0.3) + (18ms)(0.7) + 80ns] + (1-*n*)(80ns)

  Solving for *n* we get:

$$n = 1.0897435 \cdot 10^{-5} \tag{1}$$

Self Grade For #: 4

III. DEMAND PAGING

- (20 pts) Demand paging is a mechanism wherein we use a page replacement algorithm to manage the pages loaded onto the physical memory frames. Suppose that a newly created process is given 4 frames by the OS, and generates the following page references below:

$$123421351234151324 \tag{2}$$

(5 pts) How many page faults would occur with a FIFO (First-In-First-Out) page replacement policy? Place an 'X' in each box corresponding to a page fault. Place the total number of page faults under the # column.

1	2	3	4	2	1	3	5	1	2	3	4
X	X	X	X				X	X	X	X	X

1	5	1	3	2	4	#
	X	X		X	X	13

Self Grade For #: 4

(5 pts) How many page faults would occur with LRU (Least Recently Used) page replacement policy? Place an 'X' in each box corresponding to a page fault. Place the total number of page faults under the # column.

1	2	3	4	2	1	3	5	1	2	3	4
X	X	X	X				X				X

1	5	1	3	2	4	#
	X			X	X	9

Self Grade For #: 4

(c) (5 pts) How many page faults would occur with LFU (Least Frequently Used) page replacement policy? Place an 'X' in each box corresponding to a page fault. Place the total number of page faults under the # column.

1	2	3	4	2	1	3	5	1	2	3	4
X	X	X	X				X				X

1	5	1	3	2	4	#
	X				X	8

Self Grade For #: 4

(5 pts) How many page faults would occur with clock replacement/second chance replacement policy? Place an 'X' in each box corresponding to a page fault. Place the total number of page faults under the # column.

1	2	3	4	2	1	3	5	1	2	3	4
X	X	X	X				X	X	X	X	X

1	5	1	3	2	4	#
	X			X		11

Self Grade For #: 3

IV. READER-WRITER PROBLEM

- (20 pts) Reader-Writer Problem. You are given the Reader-Writer pseudocode below for accessing a shared resource: executed.

```
readerThreadRoutine() {
        lock.acquire();
        while (activeWrite > 0) {
                waitRead++;
                okToRead.wait(&lock);
                waitRead--;
        }
        activeRead++;
        lock.release();

        AccessResource(ReadOnly);

        lock.acquire();
        activeRead--;
        if (activeRead == 0 && waitWrite
            > 0)
                okToWrite.signal();
        lock.release();
}

writerThreadRoutine() {
        lock.acquire();
        while ( (activeWrite + activeRead
            + waitRead) > 0 ) {
                waitWrite++;
                okToWrite.wait(&lock);
                waitWrite--;
```

```
        }
        activeWrite++;

        lock.release();

        AccessResource(ReadWrite);

        lock.acquire();
        activeWrite--;
        if (waitRead > 0) {
                okToRead.broadcast();
        } else if (waitWrite > 0) {
                okToWrite.signal();
        }
        lock.release();
}
```

The code snippet above uses two condition variables, one for waiting readers and another for waiting writers. Suppose that all of the requests arrive in this order with a very small time interval between them. Note that none of these threads have started yet:

W1 R1 R2 R3 W2 W3 R4 R5 R6 W4 R7 W5 W6 R8 R9 W7 R10

(5 pts) In what order would the above code process the above requests? If you have a group of requests that are equivalent (unordered), indicate this by surrounding the equivalent/ unordered requests with square brackets '[]'. You can assume that the wait queues for condition variables are FIFO in nature (i.e. signal() wakes up the oldest thread on the queue). Explain your answer.

- The above code prioritizes writes over reads as implemented in the *writerThreadRoutine()* function. Therefore, the following sequence will be executed:

W1 W2 W3 W4 W5 W6 [R1 R2 R3 R4 R5 R6 R7 R8 R9 R10]

Self Grade For #: 3

(3 pts) How would the order of processing the above requests change if W1 was currently running when the other requests arrive? Explain your answer.

- As writes are still prioritized over asynchronous reads, the sequence will remain the same:

W1 W2 W3 W4 W5 W6 [R1 R2 R3 R4 R5 R6 R7 R8 R9 R10]

Self Grade For #: 2

(c) (3 pts) What will be prioritised by the code snippet above? Is it the readers or the writers? Explain your answer.

- The code prioritizes the writers first, as the condition gets trapped in a while loop and is locked until both the *activeWrite* and the *activeRead* conditions are triggered (decremented or incremented) properly to break the lock.

Self Grade For #: 3

(d) (3 pts) How does the code snippet above ensure that if activeRead > 0, then activeWrite == 0?

  –

(e) (3 pts) How does the code snippet above ensure that there can only be one writer at a time, and that if activeWrite == 1, then activeRead == 0?

  –

(f) (3 pts) Why does the reader not need to check if waitRead > 0?

  –

Self Grade For #: 4

V. INTERPROCESS COMMUNICATION

(20 pts) You are given the code template below:

```c
(1) #include <stdio.h>
(2) #include <stdlib.h>
(3) #include <string.h>
(4) #include <stdint.h>
(5) #include <stdbool.h>

(6) #include <assert.h>
(7) #include <unistd.h>
(8) #include <fcntl.h>
(9) #include <sys/mman.h>
(10)#include <sys/stat.h>
(11)#include <sys/types.h>
(12)#include <sys/wait.h>
(13)
(14)#define PAGESIZE 4096
(15)#define CTRMAX 1000000
(16)
(17)#define increment_val(ptr) (*ptr) += 1
(18)
(19)int main()
(20){
(21)    uint32_t *smem = mmap(NULL, PAGESIZE,
(22)    PROT_READ | PROT_WRITE,
(23)    MAP_SHARED | MAP_ANONYMOUS, -1, 0);
(24)
(25)    assert(smem != MAP_FAILED);
(26)
(27)    *smem = 10;
(28)
(29)    pid_t pid;
(30)
(31)    pid = fork();
(32)    assert(pid >= 0);
(33)
(34)    if (pid) {
(35)            for (int ctr = 0; ctr <
   CTRMAX; ++ctr)
(36)                    increment_val(smem);
(37)            wait(NULL);
(38)    } else {
```

```c
(39)            for (int ctr = 0; ctr <
   CTRMAX; ++ctr)
(40)                    increment_val(smem);
(41)    }
(42)
(43)    if (pid) {
(44)            printf("Data: %d\n", *smem);
(45)            int unmap_res = munmap(smem,
   PAGESIZE);
(46)            assert(unmap_res == 0);
(47)    }
(48)
(49)    return 0;
(50)}
```

- *A*. (3 pts) The code snippet above will rarely print the value 2000010 at line 45, if at all, every time the code is executed. What is the reason behind this anomaly?
  – Hi

  Self Grade For #: 4

- *B*. (6 pts) Propose a modification to the given code template to ensure that the value 2000010 is always printed at line 45 (without printing it directly!).
  – Hi

  Self Grade For #: 4

- *C*. (5 pts) If the program above were to be implemented using message passing API, would the anomaly still occur without the modification needed for shared memory API? Explain your answer.
  – Hi

  Self Grade For #: 4

- *D*. (6 pts) Present a reimplementation of the code template provided using message passing API.
  – Hi

  Self Grade For #: 4

VI. OVERUSE OF THREADS AND PROCESSES

- (20 pts) Discuss two different reasons as to why the overuse of threads/use of too many threads is bad. Discuss two different reasons as to why the overuse or processes/use of too many processes is bad. Give an example for each reason.
  – Hi

  Self Grade For #: 4

VII. PROCESS STATES

- (10 pts) Determine which state the processes below belong to given their current situation.
  – Hi

  Self Grade For #: 4

REFERENCES

[1] Talonj, *Explain morris inorder tree traversal without using stacks or recursion*, Stack Overflow, Accessed 2019-09-19, 2011. [Online]. Available: https://stackoverflow.com / questions / 5502916 / explain - morris - inorder - tree - traversal - without - using - stacks - or - recursion ? fbclid = IwAR04uuGgICHU1slLl0YC8RGnCuJAweN - 7AjLMgJ2VekA1TTo_W2KCGRfYEk.

[2] A. Kundu, *Inorder tree traversal without recursion and without stack!* Accessed 2019-09-18. [Online]. Available: https : / / www . geeksforgeeks . org / inorder - tree - traversal - without - recursion - and - without - stack / ?fbclid = IwAR2hyUkM0jKmHOmnJvwKE83Z - x2nJ5zGIT9GRECatTxegHk6KnFudKnll7Y.

[3] R. Kumar, *Features & differences in ext2, ext3 and ext4 linux filesystem*, TecAdmin, Accessed 2019-09-18, 2017. [Online]. Available: https : / / tecadmin . net / linux - extended - filesystem / ?fbclid = IwAR1R0mSK7 - rzD0yu5eW0IIKqMObTjtEXuYNr0PWu0SmwOO - HkOIhLJGobM4.

[4] Raghu, *Ext2 vs ext3 vs ext4 linux filesystem*, LinOxide, Accessed 2019-09-19, 2019. [Online]. Available: https : / / linoxide . com / how - tos / explained - in - detail - linux - ext2 - ext3 - and - ext4 - filesystem / ?fbclid = IwAR1DKfk04pC3KqJsfsvNYejTZhtjWAtISlUL9 _ 75CfRds5_NwXoiIM4fYwU.