Final Exam Online

Due May 6, 2022 at 2:45pm **Points** 100 **Questions** 17
Available May 6, 2022 at 12pm - May 6, 2022 at 2:45pm 2 hours and 45 minutes
Time Limit 150 Minutes

Instructions

COMP 3500

Introduction to Operating Systems

Final Exam

Maximum Points: 100

Exam Duration: 150 Minutes

Spring 2022

Instructions:

Write neatly and clearly. If we can't read it, you will NOT get credit even if it is correct.
Do not spend too much time on any one question – move on, and come back to it later if you have time.
If explanations are asked for, these should be brief but precise.
Some questions have multiple parts. Answer all parts of the question.

This quiz is no longer available as the course has been concluded.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	111 minutes	100 out of 100

Score for this quiz: **100** out of 100

Submitted May 6, 2022 at 1:58pm

This attempt took 111 minutes.

Question 1

0 / 10 pts

(10 Points) Question: Given there are six free memory partitions of sizes 100KB, 300KB, 500KB, 550KB, 250KB and 150KB, respectively in that order. These partitions need to be allocated to five processes of sizes 257KB, 310KB, 68KB, 119KB, 23KB in that order. Assume that the search for free partitions starts from the first memory partition. If the **NEXT-FIT** algorithm is used with **DYNAMIC** memory allocation approach, **list the sizes of free partitions (holes)** available after memory allocation to the five given processes.

Your Answer:

Sizes of free partition holes:

100KB, 43KB, 3KB, 527KB, 250KB, 150KB

Question 2

0 / 8 pts

(4+4=8 Points) Question: Say, a process of size 2048 Bytes is loaded into memory starting at address 1024. What are the valid **lowest and highest physical addresses** of the process?

Your Answer:

Lowest: 1024

Highest: 3071

Question 3

0 / 3 pts

(3 Points) Question: Consider allocation of memory to a new process. Assume that none of the existing holes in the memory will exactly fit the process's memory requirement. Hence, a new hole of smaller size will be created if allocation is made in any of the existing holes. Which one of the following statements is **TRUE**?

○ The hole created by worst fit is always larger than the hole created by first fit.

○ The hole created by first fit is always larger than the hole created by next fit.

○ The hole created by next fit is never larger than the hole created by best fit.

Correct!

[✓ The hole created by best fit is never larger than the hole created by first fit.]

Question 4

0 / 10 pts

(10 Points) Question: Assume a system comprises 16 GB physical memory with 8 KB page/frame size, and the instruction set comprises instructions that may need access to one (RET), two (PUSH A), or three (MOV A, B) memory locations to fetch the opcode and operands. Assuming that the system is configured with virtual memory:

(a) What is the **MINIMUM** number of memory frames that must be allocated to a process running in the system?

(b) What is the **MAXIMUM** number of memory frames that can be allocated to a process running in the system?

Your Answer:

(a)

Minimum number of memory frames: 1 memory frame

(b)

Maximum number of memory frames: 2096*1024 possible frames

Question 5

0 / 5 pts

(5 Points) Question: Demand paging without page replacement is not a recommended virtual memory management solution. Explain the **reason**.

Your Answer:

a page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses while balancing it with the costs of primary storage and processor time of the algorithm itself. There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults.

Therefore, page replacement algorithms are very important part of virtual memory management and it helps the OS to decide which memory page can be moved out, making space for the currently needed page. Because page replacement algorithm is important phase of demand paging

Question 6

0 / 4 pts

(2+2=4 Points) Question:

(a) Which virtual memory page replacement approach **sometimes leads to higher number of page faults** when number of **frames allocated** to process is **increased**?

(b) Which page replacement approach that does **NOT exhibit** such behavior?

Your Answer:

(a)

First in First out

(b)

Optimal page replacement algorithm gives the lowest page fault rate when the number of frames allocated to process is increased

Question 7

0 / 5 pts

(5 Points) Question: In the context of Virtual Memory management, what are **anonymous memory pages**? Is there a **need to write** them to the swap device? **Why?** Provide a brief explanation.

Your Answer:

Anonymous memory is a memory mapping with no file or devices which are backing it. Initially, an anonymous mapping only allocates virtual memory. It will return only zeros on read. when it first writes to an anonymous page, a new physical page will be allocated and filled with zeros, which will take extra memory. The implementation of swap files may vary with different operating systems, in terms of creating and using swap files as required. it also releases the drive space when it is no longer being used by a program. So, swap devices make memory management run more and more programs even with limited physical memory. So, from this we can conclude that anonymous memory pages need to write to a swap device in order to make memory more efficient.

Question 8

0 / 5 pts

(5 Points) Question: Demand paging needs support both from hardware as well as system architecture for its implementation. Briefly **explain the support** needed from hardware and from the system architecture.

Your Answer:

When a context switch occurs, the OS never copies any of the old program's pages from the disk or any of the new program's pages into the main memory. Instead, it will start executing the new program after loading the first page and fetches the program's pages, which are referenced. During the program execution, if the program references a page that may not be available in the main memory because it was swapped, then the processor considers it as an invalid memory reference. That's because the page fault and transfers send control back from the program to the OS, which demands to store page back into memory.

Question 9

0 / 6 pts

(3+3=6 Points) Question: List **TWO optimizations** that can be implemented to reduce the overhead resulting from Demand Paging.

Your Answer:

OneNand Flash Technique

Post-Pass Optimization Technique

Question 10

0 / 6 pts

(3+3=6 Points) Question: Provide an **example** for each of the following:

(a) Spatial locality

(b) Temporal locality

Your Answer:

(a) Spatial Locality

the simple traversal of elements in a one-dimensional array, from base address to the highest element would exploit the sequential locality of the array in memory.

(b) Temporal Locality

example code:

```
for(i = 0, i < 20; i++)
    for(j = 0; j <10; j++)
        a[i] = a[i]*j;
```

from the above code, a[i] is referenced frequently, with instructions like a[i] = a[i]*2 and i[i] = a[i]*3 being executed very close together.

Question 11

0 / 10 pts

(10 Points) Question: Assume we have a demand-paged memory, and the page table is held in registers whose access time is negligible. It takes **10 milliseconds to service a page fault** if an empty page is available or if the replaced page is not modified. It takes **20 milliseconds if the replaced page is modified**. **Memory access time is 100 microseconds.** What is the **maximum acceptable page-fault rate** for an

effective access time of no more than 200 microseconds? Note: Show your work to receive full grade.

Your Answer:

[CamScanner 05-06-2022 13.50.pdf (https://auburn.instructure.com/users/3643878/files/196856386?wrap=1&verifier=5OPD5eozK2asavzsCXOE39bfqBK31oRqwjFttLRs)](https://auburn.instructure.com/users/3643878/files/196856386?wrap=1&verifier=5OPD5eozK2asavzsCXOE39bfqBK31oRqwjFttLRs) **(https://auburn.instructure.com/users/3643878/files/196856386/download?verifier=5OPD5eozK2asavzsCXOE39bfqBK31oRqwjFttLRs&download_frd=1)**

If you cannot open the file please email me and i can submit my work that i have done on paper

Question 12

0 / 5 pts

(5 Points) Question: A system uses OPT policy for page replacement. A process is allocated **THREE frames**. Assume that all the frames are initially empty. What is the total number of page faults that will occur while processing the page reference string: **3, 4, 7, 2, 4, 1, 7, 6, 1, 2, 4, 7, 1, 6, 5**?

Your Answer:

The total number of page faults will be 9.

Question 13

0 / 5 pts

(5 Points) Question: Say, a system consisting of three processes sharing four instances of the same resource type. Each process needs a maximum of two resources to complete its execution. **Can the system enter a deadlock state? Why / Why not?** Provide a brief explanation.

Your Answer:

Since it is given that each process needs a maximum of two resources to complete it's execution, and we assume that the system is deadlocked then each process must be in a State with one resources allocated and waiting for another resource to complete it's execution. Since there are three processes and four resources, therefore one process will be able to obtain one additional require resource to complete it's execution. Now since one process got two resources it will complete it's execution and will return two resources which can again be used by other processes to complete their execution. As all the processes are able to execute, therefore it proves that the system can't be in a deadlocked state. Therefore the system is deadlock free.

Question 14

0 / 3 pts

(3 Points) Question: Which of the following is **NOT a valid deadlock prevention scheme**?

- ○ Release all resources before requesting a new resource.
- ○ Request all required resources to be allocated before execution.
- ○ Number the resources uniquely and never request a lower numbered resource than the last one requested.

Correct!

- [✓ Never request a resource after releasing any resource.]

Question 15

0 / 5 pts

(5 Points) Question: Will you use a monolithic kernel or microkernel to build an **OS for embedded systems** (e.g., real-time robotic systems and smart watch)? **Why?**

Your Answer:

For embedded systems, we use monolithic kernel. This is because the execution of the monolithic kernel is faster as communication between application and hardware is established using the system call. It also requires less code which further leads to fewer bugs. Whereas Microkernel designing needs more code which then leads to more bugs.

Question 16

0 / 4 pts

(4 Points) Question: Pipes are a mechanism for Inter Process Communication. Pipes have a limitation that they cannot be accessed from outside the process that created it. Which IPC mechanism **overcomes** this limitation while offering similar functionality? Explain briefly.

Your Answer:

Shared memory synchronization. In shared memory a portion of memory is mapped in to the address space of one or more processes. No method of coordinating access is automatically provided. Shared memory is a memory shared between two or more processes. Usually IPC is performed using pipes or named pipes. unrelated process communication can be performed using shared memory. Shared memory allows maximum speed and convenience of communication. shared memory is faster because the data is not copied one address space to another, memory allocation is done only once.

Question 17

0 / 6 pts

(6 Points) Question: Write a **program in C/C++** to perform the following: "**Parent process creates 5 concurrent child processes, then waits for termination of all (5) child processes prior to exit.**"

Your Answer:

```
#include <stdio.h>

#include<stdlib.h>

#include<unistd.h>


int main()

{

int pid, pid1, pid2, pid3, pid4;

pid = fork();

if(pid == 0){

    sleep(5);

    printf("child[1] --> pid = %d and ppid = %d\n", getpid(), getppid());

}

else{

    pid1 = fork();

    if(pid1 == 0){

    sleep(4);

    printf("child[2] --> pid = %d and ppid = %d\n", getpid(), getppid());

    }

    else{

        pid2 = fork();
```

```
            if(pid2 == 0){
            sleep(3);
            printf("child[3] --> pid = %d and ppid = %d\n", getpid(), getppid());
         }
         else{
                  pid3 = fork();
                  if(pid3 == 0){
                     sleep(2);
                     printf("child[4] --pid %d and ppid = %d\n", getpid(), get
ppid());
                   }
                  else{
                         pid4 = fork();
                         if(pid4 == 0){
                                printf("child[5] --> pid = %d and ppid = %d\n",
getpid(), getppid());
                         }
                     else{
                           sleep(5);
                           printf("parent --> pid = %d\n", getpid());
                           }
                     }
               }
      }
}
          return 0;
}
```

Quiz Score: **100** out of 100

This quiz score has been manually adjusted by +100.0 points.