COMP 252 Principles of Systems Software

Fall Semester 1999

Final Examination

Date: December 16, Time 8:30am – 11:30 am

Name:__SOLUTION_____Student ID:_____Email:_____

Instructions:

1. This examination paper consists of 7 pages and 7 questions.
2. Please write your name, student ID and Email on this page.
3. For each subsequent page, please write your student ID at the top of the page in the space provided.
4. Please answer all the questions within the space provided on the examination paper. You may use the back of the pages for your rough work.
5. Please read each question very carefully and answer the question clearly and to the point. Make sure that your answers are neatly written, readable and legible.
6. Show all the steps you use in deriving your answer, where ever appropriate.
7. For each of the questions assume that the concepts are known to the graders. Concentrate on answering to the point what is asked. Do not define or describe the concepts.

Question	Points	Score
1	19	
2	9	
3	10	
4	20	
5	15	
6	12	
7	15	
TOTAL	100	

1)

a) If the time quantum in a time-sharing system is *tq* and the average overhead due to swapping and context switching is *ts*, discuss the problems associated with the values:

tq --> infinity
tq <= *ts*
tq = constant (3 points each)

i) If *tq* → infinity, the algorithm degenerates to FIFO (non-preemptive) so short jobs may have to wait behind long jobs.

ii) For *tq* <= *ts* the overhead of swapping and context switching is higher than the actual time to work on the user job. The system spends much time in just bringing jobs in and out of main memory and switching jobs. This may lead to thrashing and low throughput. The system utilization will also be low.

***iii)* *tq* = constant. The problem is what value to assign to *tq*. If it is too large, the response time will be high (not interactive, see answer i) above). If it is too small, the system throughput will suffer (see answer ii)).**

b) Consider a variant of the RR scheduling algorithm where the entries in the ready queue are pointers to the PCBs (Process Control Blocks).

(i) What would be the effect of putting two pointers to the same process in the ready queue ? (3 points)

The process will get two quantum times per round.

(ii) What would be the major advantages and disadvantages of this scheme ? (3 points)

Advantage: simple way to assign more CPU time to some processes without having to change the underlying scheduling algorithm.

Disadvantage: i) the process may not be `ready' when the 2nd pointer is encountered.
ii) overhead of possibly two context switches for the process to get two quantum time.

(iii) How would you modify the basic RR algorithm to achieve the same effect without the duplicate pointers ? (4 points)

One way is as follows: For the favoured processes (those we wish to assign a longer quantum time), set the timer to a higher value just before activating the process. This scheme requires to check every job to be run and set the timer accordingly at every context switch.

2)

a) Are multiprogrammed systems usually time-shared? Why or why not? (3points)

Multiprogrammed systems maintains several jobs in main memory. This will improve system throughout by reducing the CPU wait time during context switch since it is not necessary to load the new process into main memory. Multiprogrammed systems may or may not be time-shared as even non time-shared systems will benefit from multiprogramming.

b) Are time-shared systems usually multiprogrammed? Why or why not? (3points)

Time-shared systems are usually multiprogrammed in order to save time for loading in the next job. Otherwise, the response time will suffer and it will defeat the purpose of time-sharing which is to make the system interactive.

c) Rank the following processor algorithms in increasing order of favouring short jobs: FCFS, RR, Shortest-Elapsed Time-First. (3points).

Shortest-Elapsed Time-First, RR, FCFS

3) The IBM's hard disk model Deskstar 14GXP is made of 5 platters, each capable of storing information on both surfaces. It is formatted to contain in average 200 sectors per track. Assume that surfaces are numbered from top to bottom starting with 0, and cylinders are numbered from outermost to innermost starting with 0. Disk has 13085 cylinders.

a) Given the sector and block size is 512Bytes, calculate the maximum file size which can be achieved without disk arm movement. (5 points)

Answer: this is equal to the cylinder capacity, i.e. 10*200*512= 1024 * 10^3 bytes

b) Calculate the surface number, cylinder number and sector number where physical block number 9090 is located on the disk. (5points)

Answer: Cyl = (int) 9090/2000 = 4, head = (int) 1090/200 = 5, sec = 90

4) Below is the listing of a short assembly language program for a computer with 512-byte pages. The main program is starts at virtual address 1020, and its stack pointer is initialized at virtual address 8192 (the stack grows towards 0).
   a) Give the page reference string generated by this program. Each instruction occupies 4 bytes (1 word), and both instruction and data references count in the reference string. Take care that "push" instuction involves decrementing of the value of stack pointer and storing the data afterwards. (16 points)

      i) Load word at virtual address 6144 into register 0.
      ii) Push register 0 onto the stack.
      iii) Call a procedure at 5120, stacking the return address.
      iv) Subtract the immediate constant 16 from the stack pointer.
      v) Compare the result of previous operation to the immediate constant 4.
      vi) Jump if equal to 5152.

   a) If this process is allocated 3 physical frames, how many page faults will occur during the execution of the previous sequence? Assume that initially the process is not loaded in the physical memory and that the page replacement policy is LRU. (4 points)

Answer:

a)

i)	instruction fetch from page	(int) 1020/512 = 1	//page fault	1, empty, empty //2pt
	data load from page	(int) 6144/512 = 12	//page fault	1,12, empty //2pt
ii)	instruction fetch from page	(int) 1024/512 = 2	//page fault	1,12,2 //2pt
	stack push into page	(int) 8188/512 = 15	//page fault	15,12,2 //2pt
iii)	instruction fetch from page	(int) 1028/512 = 2	//	15,12,2 //2pt
	push the return address in page	(int) 8184/512 = 15	//	15,12,2 //2pt
iv)	fetch instruction from page and subtract SP-4 in register	(int) 5120/512 = 10	//page fault	15,10,2 //2pt
v)	instruction fetch from page	(int) 5124/512 = 10	//	15,10,2 //1pt
vi)	instruction fetch from page	(int) 5128/512 = 10	//	15,10,2 //1pt

b)

F1 F2	1	1 12	1 12	15 12	15 12	15 12	15 10	15 10	15 10
F3 Page fault	x	x	2 x	2 x	2	2	2 x	2	2

5 page faults = 4 points, don't count the first one and deduct 1 pt. if each of the subsequent four page faults is wrong.

5) Three processes, A, B, and C are currently loaded into physical memory. Their current memory requirements (in bytes) are as follows. (15 points)

Process	Code Segment	Data Segment	Stack Segment
A	492	438	2009
B	4034	1030	610
C	8900	914	1120

a) The OS supports paging with a page size of 512 bytes. How much physical memory is wasted for each of the above processes due to fragmentation? Explain your answer. (6points)

Answer:
Process A: (512-492) + (512-438) + (2048-2009) = 133 //1.5pt
Process B: (4096-4034) + (1536-1030) + (1024-610) = 982 //1.5pt
Process C: (9216-8900) + (1024-914) + (1536-1120) = 842 //1.5pt

Each segment is allocated a whole number of pages and it starts at the page boundary. The last page for each segment might not be full. (1.5 pt)

b) What kind of fragmentation is encountered above? (3points)

Internal fragmentation.

c) Suppose that the OS (+ hardware) is modified to support paged segmentation (with the same page size of 512 bytes), such that each segment above can be loaded separately, how much memory is wasted due to the fragmentation? Explain your answer. (6points)

Process A: 492 + 438 + 2009 = 2939 wasted space 3072-2939 = 133 // 1.5 pt
Process B: 4034 + 1030 + 610 = 5674 wasted space 6144-5674 = 470 // 1.5 pt
Process C: 8900 + 914 + 1120 = 10934 wasted space 11264-10934 = 330 // 1.5 pt

Now, each individual segment is no longer alligned at the page boundary. Segments can be placed end to end, so only the last page of the last process may not be full. (1.5 pt)

6) Two CS/CPEG students Maggie and BoBo are having a discussion about the comp252 matters.
   a) First they discuss I-nodes. Maggie maintains that memories have gotten so large and so cheap that when a file is opened, it is simpler and faster just to fetch a new copy of the I-node into the I-node table, rather than to search the entire I-node table to see if it is already there. BoBo disagrees. Who is right and why? (6 points)
   b) Then, they discuss swap space implementation. Maggie maintains that the image of the whole process has to be stored in the swap area before process starts execution and that whole paging should be done from the swap space only. BoBo claims that swap area should be used for the data pages only. She also claims that program text should not be saved in the swap area but it should be paged directly from the executable file in the file system whenever it is needed. According to the today's state of the OS and computer hardware technology, whose approach is more efficient and why ? (6 points)

Answer:

a) BoBo is right. There must exist unique I-node per each file in the system to keep information about the file layout. Otherwise, different processes can write into the file and increase its length, and this information will be kept in different places .

b) BoBo is right. Today, the memory is large and lot of file system data structures can be kept in the main memory instead of on the disk. Therefore, it is more efficient to re-read the page from the file system than to write it to the swap space and re-read it from there. Also, the swap space is smaller since it contains the data pages only.

7) a) A section of the File Allocation Table (FAT) for a file system is given below. Pointer to the list of free blocks points to block 1. a) How many files are represented in this FAT? b) What is the length (in the number of blocks) of each file and the free space? c) What is the lay-out of each file (i.e. which disk blocks constitute the file) ? (7 points)

FAT Entry no.	FAT
0	15
1	0
2	EOF
3	EOF
4	8
5	11
6	EOF
7	4
8	EOF
9	EOF
10	12
11	3
12	14
13	6
14	7
15	9

Answer:
Free space : 1,0,15,9 // 1pt
File 1 – 1 block: 2 // 1.5pt
File 2- 3 blocks : 5,11,3 //1.5pt
File 3 – 2 blocks: 13,6 //1.5pt
File 4 – 6 blocks: 10,12,14,7,4,8 //1.5 pt

b) Consider the organization of a UNIX file as reperesented by I-node. Assume that there are 12 direct block pointers, and a singly, doubly, and triply indirect pointer in each I-node. Further, assume that the system block size and the disk sector size are both 8K. The disk block pointer is 32 bits long. Assuming no information other that the file I-node is already in main memory, how many disk accesses are required to access the byte in position 13,423,956 ? (8 points).

Answer: (int) 13423956/8192 = 1638 is the position of the block within the file. // 4pt
Since the indirect block pointer points to the block with 2048 pointers, the required pointer is placed in the first indirect block. Therefore two more disk accesses are needed, one for the block with pointers, and another for the data block. // 4pt