34 ~~44~~
~~32~~
/44

CS570 Summer 2011 Exam #2

6/21/2009

Name [redacted]

Username masc [redacted]

1. $\frac{6}{6}$ 2: $\frac{4}{4}$ 3: $\frac{0}{4}$ 4: $\frac{3}{4}$ 5: $\frac{6}{6}$ 6: $\frac{4}{4}$ 7: $\frac{6}{6}$ 8: $\frac{3}{6}$ = 40

please → look back of this page also

1. In a system that has 8 GByte RAM, 1MByte (1048Kbyte) virtual memory (for page storage), and can handle up to 30,000 processes simultaneously, is the amount of disk space available adequate for this system (explain)? Include an expression for the worst-case disk space requirements and explain. Because we know the disk space requirements is dependent on number of processes, address space in virtual memory, and the space in RAM we know in worst_case the amount of space for disk we need will be

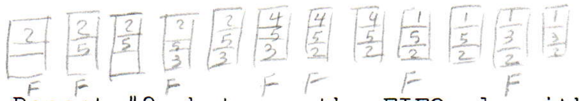
$$\left[\underbrace{(\text{number of processes})}_{n}\underbrace{(\text{address space in Virtual memory})}_{v}\right] - \underbrace{\text{memory in RAM}}_{r}$$ ✓

worst case

=> $nv - r$ can be the worst-case expression. Here we have $n = 30000$, $r = 8$ GByte, $V = 1048$ Kbyte => $(30000)(1048) - 8(2096) = 31423232$ kbyte

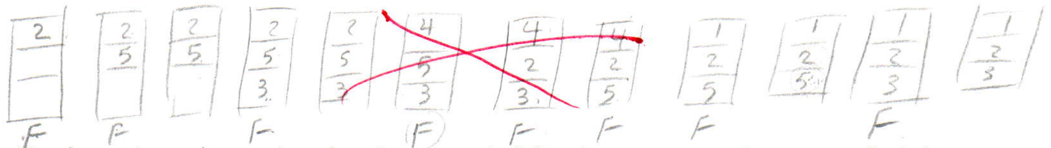
2. Given an O/S which has divided all of fast memory into 3 frames, and if this O/S uses the LRU algorithm for page replacement/swapping, then how many page faults are generated if the following sequence of pages are requested for use (show your work on back!): 7 ✓
2 5 2 3 5 4 2 5 1 2 3 2

LRU



3. Repeat #2, but use the FIFO algorithm for page replacement/swapping instead. How many page faults? 8

FIFO



4. Each entry in a typical page table has several control bits to enable the h/w and operating system to manage the pages with. Name the correct control bits for each below:
   1. This page table entry is valid: present bit ✓
   2. The page at this entry has been written to: Modified bit ✓
   3. The page at this entry may be read fm/written to: read bit/ ~~write~~ bit
   4. The page at this entry has been read fm/written to: ~~modified bit~~ OK

protection
refrencal

5. Name the six file types implemented in Unix. Give a description/example of each:

1) Regular files ✓: such as .exe, these files contain user data and may be more than 1 block

2) Directory files ✓: system files that help managing file system and are the only files with children. we can't remove a nonempty Directory file and the directory file must know about the files it contains for example it must know the name of the files it contains.

please Look at the back of this page

Question 5:

3- pipe files : files that are connecte via pipe, these files don't have children and can only have one block. also no user information

4- link files : files that link other files together, these files also don't have children and only have one block. don't contain user information.

5- Device drivers: files that are device drivers and don't contain user information

6- sockets: these files do not have user information and can only be one block.

Question 1:

Yes the amount of disk space is enough becuse in reality we rarely face the worst case senario in which the system has to handle n processes at the same time.

6. Some operating systems provide a system call, "rename()", to give a file a new name. Is there any difference between using this call to rename a file and just copying the file to a new file with the new name, followed by deleting the old one (include a detailed explanation of why/how)

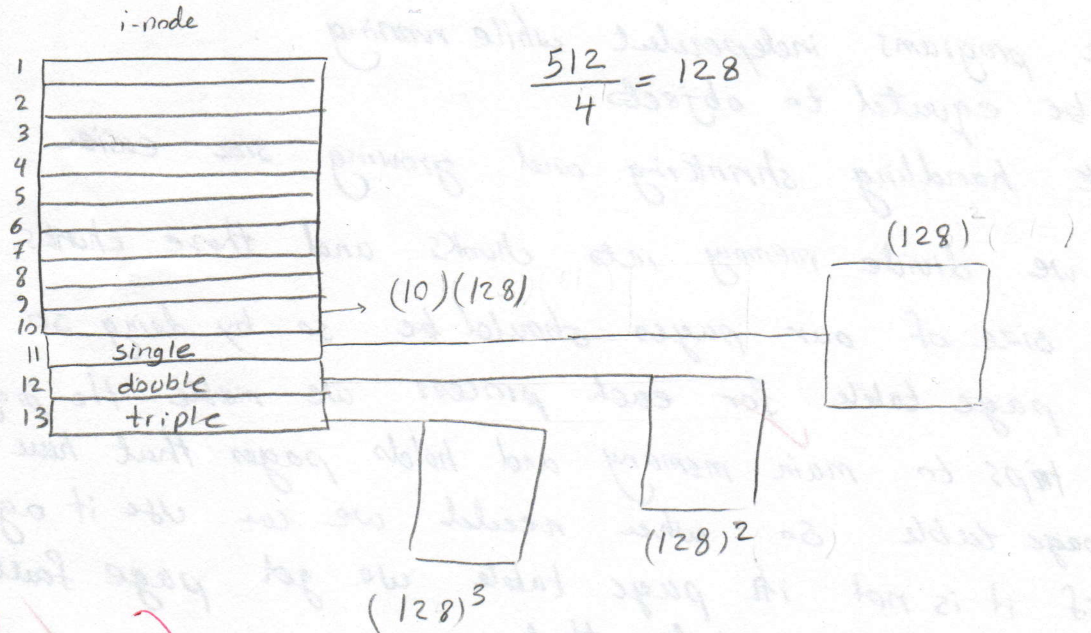
Yes, there is a difference. To rename a file, we don't change anything thus the creation data and the date of last modified stays the same as before. But, to copy the file to a new file, we change the creation date and the date of last modified both to the creation and last modified date for the new file. Also when we want to create a new file, if disk is full, copying fails.

7. List three advantages of using page-segmented memory over segmented memory. Be specific/detailed and explain why it's an advantage:

In pag-segmented memory we have combined paging which is transparent to programmer with segmentation which is visible to a programmer. Meaning that we put fixed-size pages into segments that can vary in length so we are having the best of both worlds, we may get a lower compilation speed but we only compile once so we improve run time speed which is always better since we almost always run more than one time. please see back of this page

no, the other way around

8. A UNIX filesystem has 512 byte blocks and 4 byte disk addresses. What is the maximum filesize assuming inodes have 10 direct, one single, one double, and one triple indirect addresses each?



$$\frac{512}{4} = 128$$

$512 * \left(10 \times (\cancel{128}) + (128) + (128)^2 + (128)^3 \right) = 2114944$ bytes is the maximom file size

-2

-1

7- Also in page-segmented memory we

virtual address

segment #	page #	offset

page

present bit	modified bit	other controls	page frame

segment

control bits	length	segment base

→ Here also in segment section we don't need modified bit or present bit because we already have that in page section.

in segmented memory we get the following:

- segments lends themselfs to protection
- they make programs independent while running.
- they can be equated to objects
- they make handling shrinking and growing size easier

in paging we divide memory into chunks and these chunks decide what the size of our pages should be so by doing so and having a page table for each process we make the program make less trips to main memory and holds pages that have been frequently used in page table so when needed we can use it again and only if it is not in page table we get page faults and go check main memory for that.

So combining segmentation and paging we get the benefit of both these two methods speed up our process and optimize the performance at runtime since runtime can happen more than once eventhough we get the drawback of slower compile time