COSC 4330 FINAL DECEMBER 10, 2001

THIS EXAM IS CLOSED BOOK. YOU CAN HAVE ONE PAGE OF NOTES. UH EXPELS CHEATERS.

1. A computer has 64 bit addresses and a virtual memory with a page size of 4 kilobytes. (2×5 points)

(a) How many bits are used by the byte offset? __12_____ **bits**

(b) What is the maximum number of pages per process? __2^{52}____ **pages**

2. A 32-bit Berkeley UNIX file system has a block size of 16 kilobytes. How many blocks of a given file can be accessed :

(a) using the block addresses stored in the i-node? (2 points) _____ 12 **blocks**

(b) with one level of indirection? (2 points) _____ 16K/4 = 4096 **blocks**

(c) with two levels of indirection? (2 points) _____ 256K -4096 – 12 **blocks**

Explain your answer to point (c) above. (4 points)

Since we have a 32 bit file system, the maximum file size is 4 Gigabytes. The maximum number of 16 KB blocks in a file is thus 4GB/16KB = $2^{32}/2^{14}$ = 2^{18}= 256K.

3. The access time to the main memory of a computer is 60 nanoseconds and its disk access time is 8 milliseconds. What is the mean access time to the *virtual memory* of the computer when its page fault rate is one page per ten thousand references? (10 points)

Hint: Do not use a calculator; round up instead**.**

T_{access} = 60 ns + 1/10,000 8 ms = 60 ns +1/10,000 8,000,000 ns = 860 ns

4. Show how deadlocks can be prevented without denying mutual exclusion, without allowing preemption and without forcing processes to acquire all their resources at the same time. (5 points) What is the major limitation of this approach? (5 points)

We can also prevent deadlocks by requiring all processes to acquire their resources in the same linear order, thus denying the circular wait condition. The technique does not apply to message passing systems as it would require all messages to go in the same direction.

5. Describe in some detail the Mach page replacement policy. (10 points)

Mach divides its main memory into a pool of page frames shared by all processes and one global queue from which pages can be reclaimed.

The pool of page frames is managed according to a Global FIFO policy but pages expelled from the pool by the FIFO policy are given what essentially amounts to a *second chance.* Instead of being immediately expelled from the main memory, they go at the end of the global queue and their *valid* bit is reset to *zero*.

Whenever a page fault occurs, the page fault handler looks first at the global queue to see if the missing page is cannot be found there. If this is the case, the page is returned to the FIFO pool. If another page needs to be expelled from the FIFO pool to make space for the returning page it will go at the end of the Global queue and another page put at the end of the global queue.

6. *Advantage and disadvantages: you will get no credit if you answer mentions a disadvantage when an advantage is asked and vice versa.* (4×5 points)

(a) What is the major advantage of *inverted page tables*?

The size of inverted page tables is proportional to the size of the main memory and not to that of process address spaces.

(b) What is the major disadvantage of using the *valid bit* to simulate an non-existent *page referenced bit?*

It introduces additional context switches as any page whose valid bit was reset to zero by the kernel will have its valid bit set back to one by the kernel the first time it is referenced again..

(c) What is the major disadvantage of the *VMS page replacement policy?*

It assumes that we can accurately select the size of the fixed partition assigned to each new process.

(d) What is the major advantage of *contiguous allocation* for files on disk?

It provides the fastest possible access to the file..

7. What is the difference between the *dirty bit* and the *page referenced bit*? (5 points) Which one is more important than the other and why? (5 points)

The dirty bit indicates whether the apger has been modified since the last time it was brought into main memory ("dirty") or has remained unchanged ("clean") while the page-referenced bit indicates whether the page has been rencently referenced or not.

Many architectures lack a page-referenced bitbut none lacks a dirty bit.

8. Java implementation of monitors does not support *named conditions*: whenever a **`synchronized`** method does a **`wait()`**, it cannot specify a condition to wait on. In exchange, Java offers a **`notifyAll()`** primitive that notifies all waiting methods that the state of the system has changed but without telling what happened. Complete the following solution to the bounded buffer problem without using named conditions. (4×5 points)

```
Class Bounder_Buffer {
   //private declarations
   private int buffer_size, nfullslots;
   // monitor procedures
   public void synchronized put(Entity item) {
        _ while (nfullslots == buffersize)  wait(); _________;
        put_item_in_the_buffer();
        nfullslots++;
        __ notifyAll(); ____________________________________;
   } // put

   public void synchronized get(Entity item) {
        _ while (nfullslots ==0)  wait(); ___________________;
        get_item_from_the_buffer();
        nfullslots--;
        __  notifyAll(); ___________________________________;
   } // get

// constructor
   Bounded_Buffer(int size) {
        nfullslots = 0;
        buffer_size= size;
   } //constructor

} // monitor Bounded_Buffer
```