UMass Boston CS 444 Spring 2025
Review for Test1

1. TAKING TEST1:

Please show your id and turn off any digital device(s) you have brought with you. No notes are allowed. I'll write the last four of you UMB id on the test.

The two of you will be given a page with your (randomized questions) test1 on it. You will also be given printed pages of any figures referred to in your questions.

Your test1 will have 4 sections:

(2 mins) Q1 Hello… and what did you find interesting is Chapter n? Chapter n? Lower last four in UMB id has the first chapter listed and n = {1,…,5}.

(8 mins) Q2 8 True/False…

(8 mins) Q3 C Code questions:
   (4 mins) a)... and
   (4 mins) b)…

(8 mins) Q4 OS Methods questions:
   (4 mins) a) … and
   (4 mins) b) …

The OS Methods questions are asking you to relate information in your user program to abstractions about how the operating system is working to assist running your program or helping other users run theirs.

Q1 counts 8 points (0 points if neither student has an answer), Q2 counts 32 points (4 points per T/F), Q3a, Q3b, Q4a and Q4b count 15 points. You and your test partner will get the same score. On Q1, you each give your own prepared answer on the chapter that you have randomly received.

You will write your answers on the whiteboard. You will be able to discuss and make notes on the whiteboard and refer to them as you answer. For each question, you will give your answer verbally when you are ready.

The test takes <u>about</u> 25 minutes. Time goes quickly. I will bring a clock and note the start time for each question. Please keep an eye on how time is progressing and give your answers within the time limit for each question. I'll give you your score.

2. WHAT IS INCLUDED. The test1 covers reading Chapters 1-5 from the textbook, slides chapters 1-5, huffman.pdf, and hamming.pdf, hw1, hw2, and proj1.

Hwk1 covered huffman code prefix code tree and reading codes, metric prefixes and ASCII codes, scheduling algorithms and priority inversion. Hwk2 covered hamming code example, page table examples and drawing memory.

These are the chapters studied:
Chapter 1. Introduction.
Chapter 2. Processes and Threads.
Chapter 3. Memory Management.
Chapter 4. File Systems.
Chapter 5. Input/Output.
For Q1, find something you were interested to learn in each chapter. **You need to know the name of each chapter because test1 gives you only the chapter number.**

These are the C code figures:
 Fig 1-19, p55, A stripped down shell. True is defined as 1.
           p75, C code in text for pointers
 Fig 1-30, p77, Compiling – try command “which gcc”, is it /usr/bin/gcc?
 Fig 2-14, p108, Program creating threads.
 Fig 2-23, p123, A proposed solution to the critical region problem.
 Fig 2-27, p129, Producer Consumer with fatal race condition.
 Fig 2-28, p131, Producer Consumer with semaphore.
 Fig 4-6, p271, A simple program to copy a file.
 We also studied endian.c.
 You need to know how pointers and malloc work from Kernighan and Ritchie.
 You need to know how you wrote your C code for the huffman algorithm.

These are the basic Chapter 1 figures that summarize the textbook:
 Fig 1-1, p2, Where the Operating System fits in.
 Fig 1-5, p12, A Multiprogamming system.
 Fig 1-6, p21, Some of the components of a simple personal computer.
 Fig 1-7, p22, a) 3-stage pipeline, b) superscalar CPU
 Fig 1-9, p25, A typical memory hierarchy.
 Fig 1-10, p28, Structure of a Disk Drive.
 Fig 1-11, p32, a) Steps in starting an I/O device and getting an interrupt. b) Interrupt processing.
 Fig 1-13, p41, Process tree
 Fig 1-17, p52, The steps in making the system call read(fd, buffer, nbytes).
 Fig 1-18, p54, Some of the major POSIX system calls.
 Fig 1-20, p57, Processes have three segments.
 Fig 1-23, p62, The corresponding Win32 API calls.
 Fig 1-31, p81, Principal metric prefixes.

These are OS method figures:
 Fig 2-11. Per process vs per thread items, p104.
 Fig 2-2. p93, States of a process.
 Fig 3-12, p202, A Translation Lookaside Buffer to speed up paging.
 Fig 3-9, p195,Page Table.
 Fig 3-10, p197, Operation of MMU.
 Fig 4-8, p274, Hierarchical Directories.
 Fig 4-15, p284, Inodes.
 Fig 5-14, in section 5.3.3, p363. With/without a standard driver interface.

3. Chapter 1: Introduction.
T/F examples:
Virtual memory cannot be bigger than physical memory.

When the OS runs in user mode, it has access to all the hardware and can execute any instruction the machine is capable of.

The Minux3 microkernel is about 15,000 lines of C code and 1400 lines of assembler. (p67)

Windows and Linux are in the 10s of millions of lines of code. (p3, Windows > 50 million, Linux > 20 million)

The memory hierarchy has four layers. (p25)

ANS: F, F, T, T, T.

OS Method question:
Q1. Figure 1-17, p52, Explain the 11 steps in making a read system call.

Q2. Figure 1-17, p52, Explain the 9 steps in making an exit system call.

4. C code questions
Figure 1-19 A stripped down shell, p55:
Q1. Notice the three system calls in the figure: exec, fork and exit. What type of system calls are they? (p55-56)

Q2. Notice the three system calls in the figure: exec, fork and exit. What does exec do? (p55-56)

Q3. Notice the three system calls in the figure: exec, fork and exit, what does fork do? (p55-56)

Q4. Notice the three system calls in the figure: exec, fork and exit, what does exit do? (p55-56)

Q5. After a fork, the parent and child share a copy of memory: when does the OS give the child process its own address space? (p55-56)

5. Chapter 2: Processes and Threads.
T/F:
PSW means Program Status Word.

When two processes are reading or writing some shared data the final result does not depend on who runs precisely when.

Locking the memory bus has the same effect as disabling interrupts.

Answers: T,F,F:
See page 126 in textbook:
"It is important to note that locking the memory bus is very different from disabling interrupts. Disabling interrupts than performing a read on a memory word followed by a write does not prevent a second processor on the bus from accessing the word between the read and the write. In fact, disabling interrupts on processor 1 has no effect at all on processor 2. The only way to keep processor 2 out of the memory until processor 1 is finished is to lock the bus, which requires a special hardware facility (basically, a bus line asserting that the bus is locked and not available to processors other than the one that locked it)."

6. Chapter 3. Memory Management.
T/F:
NULL is the integer three, as defined in stdio.h.

CPU utiliization is a function of the number of processes in memory.

Race conditions can be solved by the use of critical regions and mutual exclusion.

For int *pa;, you can write pa=&a[0] and pa=a.

ANS: Consult Kernighan and Ritchie. See p176 for stdio.h contents, Also see textbook p75 for pointer handling. Also see textbook, p96 for CPU utilization. See textbook pp119-124 for race condtions.

Answers: F, T, T, T

Problem on Average memory access time.
See pages 200-203, section 3.3 Speeding up paging.
Memory system: Let's use a computer that has L1 and L2 caches, translation

lookaside buffer (TLB) and page table (PT).
Action for MMU: Data at an address is needed.
How to calculate average memory access time? Statistics are collected by the OS, so we know how often certain steps occur.
MMU steps:
1. MMU simultaneously looks up the data in L1 and L2 cache.
1.1 If the data is in L1 cache, the MMU spends no time for this step, which occurs with probability **0.9.**
1.2 The data is in L2 cache. MMU takes X units of time to retrieve it, which occurs with probability of **0.09.**
1.3 If the data is in L1 or L2 cache, skip steps 2, 3 and 4. Done. If the data is not in either, the MMU proceeds to Step 2, with probabilty **0.01.**

2. MMU simultaneously queries the TLB and the PT. Assume that Z > Y > 0.
2.1 MMU gets the page frame number from the TLB, which takes Y units of time, and occurs with probability 0.98.
2.2 MMU looks up the page frame number from the PT, which takes Z units of time, with probability 0.02.
2.3 If step 2.1 succeeds, step 2.2 aborts. Either way, the MMU has the page frame number and it goes to step 3.

3. If a page fault occurs, the OS kernel needs U units of time to bring the page from disk to RAM. This happens with a probability of 0.9999. At this point, the page we need is in RAM.

4. Finally, the MMU delivers the data from RAM to L1 cache, taking V units of time.

Now calculate the average memory access time by looking at each set of steps and adding them up.

Answer: to be worked out in class.

OS Method questions:

Q1. In Figure 1-9, p25, Typical Memory Hierarchy, there are four levels with sizes and access times. What steps from the above average memory access time problem does the MMU do to retrieve an instruction’s data in register level access time?

Q2. In Figure 1-9, p25, Typical Memory Hierarchy, there are four levels with sizes and access times. What steps from the above average memory access time problem does the MMU do to retrieve and instruction’s data in cache level access time?

Q3. In figure 1-9, p25, Typical Memory Hierarchy, there are four levels with sizes and access times. What steps from the above average memory access time problem does the MMU do to retrieve an instruction’s data in main memory level access time?

Q4. In Figure 1-9, p25, Typical Memory Hierarchy, there are four levels with sizes and access times. What steps from the above average memory access time problem does the MMU do to retrieve an instruction’s data in non-volatile storage access time?

Answers:
register level: step 1.1;

cache level: step 1.2;

main memory level:
option 1: steps 1.3b – proceed to step2, 2.1, 2.3a – got it in 2.1, 3b -didn’t need to go to disk, and 4 or

option 2: steps 1.3b, 2.2, 2.3b – got it in 2.2, 3b -didn’t need to go to disk and 4;

non-volatile storage: steps 1.3b -proceed to step2, 2.3 get page frame number, 3 – page fault, so go to disk, 4 deliver.

7. Chapter 4. File Systems.
Be sure to look at Figure 4-6. A simple program to copy a file, p271. Understand pp270-272 and be able to explain when the code is doing a system call and what the operating system is doing for the code.

Study I-nodes, p284-285.

T/Fs:
The Makefile does not have an extension in its file name.

As shown in Figure 4-21, the virtual file system has two interfaces, one to user process running code and one to physical file system(s) with buffers.

Once block size is chosen by the designers of the file system, a method of keeping track of free blocks can be selected from common choices of linked-list or bitmap.

ANS:
First one is true – no extension.
Second one is true – Be sure to know Figure 4-21 and the text that talks about it.
Third, True. Review the figures in section 4.3.2, especially what is the i-node. Be sure you see why the two ways of implementation work.

8. Chapter 5. Input/Output.
Get the idea of I/O devices from section 5.1.1 and the list in Figure 5-1.

T/F:
If execution of instructions can be out of order and interrupted when partially complete, the only type of interrupt possible is precise interrupts.

Ans: See Section 5.1.5 pp347-351. Figure 5-5 illustrates the interrupts that can happen. Know precise and imprecise interrupts in the text and in Figure 5-6. The statement is false. Both are possible.

Another T/F:
No matter whether a CPU does or does not have memory-mapped I/O, it needs to address the device controllers to exchange data with them.

Ans: See p342. This is true.

Chapter 5.6 User Interfaces.
In a layered diagram, with hardware at the bottom, the U/I would be at the top. There is a big difference between command-line access and access via a window. Both ways select programs to run. Any OS will make processes and threads to run the program. For command-line access, the shell handles the U/I hardware. For access via a window, the windowing support software handles the U/I hardware. Both softwares have to issue system calls to move from user space to kernel space and access the OS resources needed.