Inclusion of a question or topic in this document does not guarantee it will appear on the final exam. In the same light, omission of a question or topic does not guarantee it will not appear on the final exam. In addition, the format of a question may change. A question that is free answer in this document may appear as a true/false on the final exam. The final exam could be100 True/False questions, with 99 of them False (don’t bet on this though).

**You may use two 3-inch by 5-inch index cards for notes on the final exam.**

**beavalloc() – everyone’s favorite – some possible questions. No eager splits.**

Assume the following data structure, as a **doubly linked list**.

struct block\_list {

uint8\_t free; // TRUE or FALSE

void \*data;

size\_t capacity; // the total bytes represented in the data field

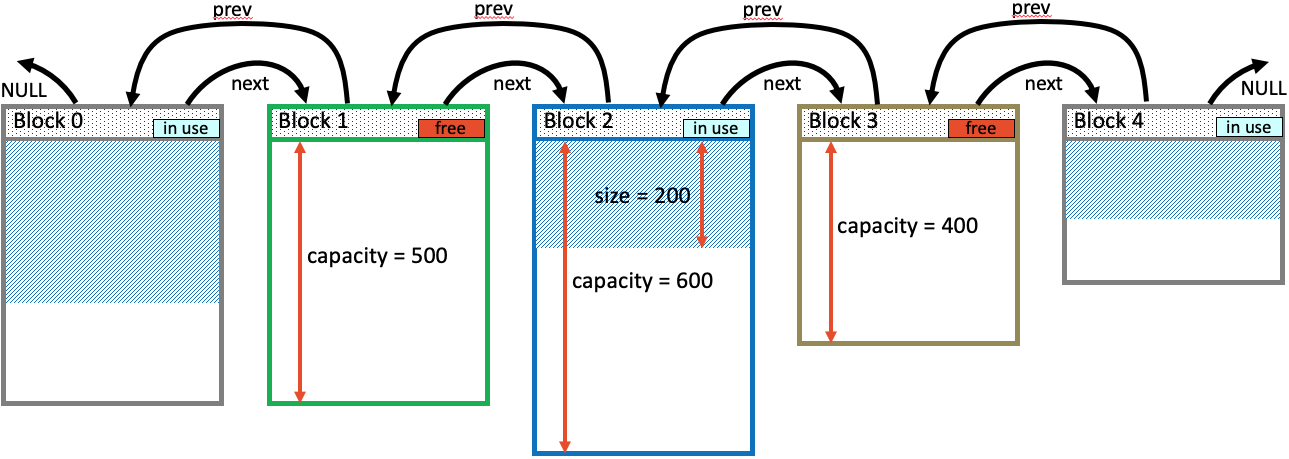
size\_t size; // the number of bytes requested by the user

struct block\_list \*prev;

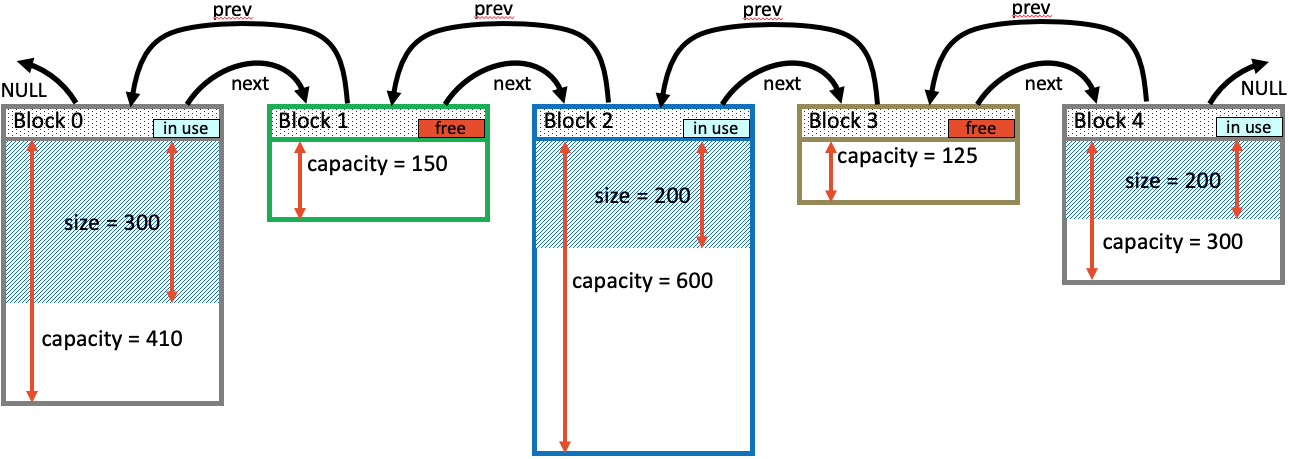
struct block\_list \*next;

};

Think about making some diagrams.

1. Write the C code to implement a “first fit” algorithm when a user calls beavalloc(). See question 11.
2. Write the C code to implement a call to beavfree(). Don’t worry about coalescing.
3. Write the C code to coalesce blocks when beavfree() is called. Assume that beavfree() is called and the block to be freed has a free block on each side of it. See question 9.
4. Write the C code to split a large block into 2 blocks when beavalloc() is called. One block should be split such that capacity = size and other (new) block gets the remaining capacity. See question 10.
5. Write the C code for beavrealloc(). A call to beavrealloc() should only allocate a new block if there is not a block that can accommodate the requested size.
6. Write the C code for beavalloc\_reset().
7. Write the C code for beavcalloc().
8. What is the difference between sbrk() and brk()?
9. Assume the following configuration of data blocks in the heap. What is the coalesced outcome if Block 2 is freed? Draw it, label the sizes of things.

**Figure 1: Coalesce blocks.**

1. Assume the configuration of data blocks in the heap shown in figure 2. What is the outcome if a call to beavalloc() results in Block 2 is being split? Assume that the call to beavalloc() requests 300 bytes. Draw it, label the sizes of things. 

**Figure 2: Allocated blocks in the heap.**

1. Assume the configuration of data blocks in the heap shown in figure 2. What is the outcome if a call to beavalloc() uses a **first fit** algorithm and the request is for 120 bytes. Draw it, label the sizes of things.
2. Assume the configuration of data blocks in the heap shown in figure 2. What is the outcome if a call to beavalloc() uses a **best fit** algorithm and the request is for 120 bytes. Draw it, label the sizes of things.

**Week 1 material**

1. For our class, what is a process?
2. How is a process different from a program?
3. Is it possible to have multiple instances of the same program running at once?
4. In our class, what does “direct execution” mean?
5. What are some potential problems with “direct execution?”
6. What is the difference between “user mode” and “kernel mode?”
7. When the OS switches out process A so process B can run, we call that a \_\_\_\_\_ \_\_\_\_\_\_\_.
8. What is *cooperative* multitasking?
9. What are some problems with cooperative multi-tasking?
10. Describe how with cooperative multi-tasking differs from true multitasking.
11. For our class, what does PCB stand for?
12. What data are kept in a PCB?
13. Each process can be identified by a unique \_\_\_\_\_\_\_.
14. The basics of the Von Neumann model of computing include what steps?
15. What does CPU virtualization accomplish?
16. How can 2 different programs running have a pointer to the "same" address that contains different values?
17. For our class, you can think of a thread as \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
18. The basic concept used to vitalize a CPU is \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
19. Describe what is meant by “process address space.”
20. In a simplified view, a process can be in one of three states:
21. When a process calls fork(), what is the return value for the parent process (on success)?
22. When a process calls fork(), what is the return value for the child process (on success)?
23. When a process calls fork() (successfully), which process will run first, parent or child?
24. When a parent process calls wait(), it will
25. When a process calls the exec() function, it will
26. To execute a system call, a program must execute a special \_\_\_\_\_\_\_\_\_ instruction.

**Week 2 material**

1. What are the 2 basic ways to create a process?
2. Draw the stat transition diagram for the 3 fundamental states of a process in an OS.
3. Define Workload
4. Define Scheduler
5. What are some of the scheduling performance metrics we’ve discussed?
6. Describe the FIFO scheduling algorithm.
7. Describe the shortest job first scheduling algorithm.
8. Describe the shortest time-to-completion first scheduling algorithm.
9. Describe the round robin scheduling algorithm.
10. What is an obvious problem with a FIFO scheduler?
11. True or False: For minimizing average turnaround time (with no preemption) - SJF is provably optimal.
12. Describe a “preemptive scheduler.”
13. What does “response time” measure? (give me an equation)
14. An OS Must support two job types with distinct goals

“**interactive**” programs care about **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

“**batch**” programs care about \_\_\_\_\_\_\_\_\_\_\_\_

1. What is MLFQ and how does it work?
2. Is MLFQ a “learning” algorithm?
3. What can occur when using MLFQ when there are too many interactive jobs?
4. Describe the lottery scheduler algorithm.

**Week 3 material**

1. A Unix user process is made up of 4 segments. Describe the segments.
2. Multi-programming has 4 goals. What are they? Briefly, describe them (1 or 2 sentences).
3. Describe internal fragmentation. Diagram it.
4. Describe external fragmentation. Diagram it.
5. What segments of a Unix process are static and which ones are dynamic?
6. From which Unix process segment are automatic variables allocated?
7. What happens to automatic variables within a function when it returns?
8. Based on the code below, which variables are allocated from the stack?

main () {  
 int A = 0;  
 foo (A);  
 printf("A: %d\n", A);  
}

void foo (int Z) {  
 int A = 2;  
 Z = 5;  
 printf("A: %d Z: %d\n", A, Z);  
}

1. Using the following, identify in which segment each identifier is placed.

int x;

int main(int argc, char \*argv[]) {

int y;

int \*z = malloc(sizeof(int));

}

|  |  |
| --- | --- |
| **Address/identifier** | **Segment** |
| x |  |
| main |  |
| y |  |
| z |  |
| \*z |  |

1. What is the major problem with time sharing memory?
2. What is static relocation for managing memory for an OS?
3. What are some disadvantages of static relocation?
4. Dynamic relocation introduces a new piece of hardware. What is it and what does it do?
5. True or False: Multiple processes often share a value for a base register.
6. What entity should do the translation of addresses with base register?
   1. Process
   2. OS
   3. H/W
7. What entity should modify the base register?
   1. Process
   2. OS
   3. H/W
8. What register in a MMU is used to prevent one process from accessing the address space of another process?
9. Are the MMU registers kept as part of the PCB?
10. What is the disadvantage of using the base and bounds registers for processes?
11. What is process segmentation?
12. List some advantages of process segmentation.
13. List a disadvantage of process segmentation.

**Week 4 material**

Extra Credit: The Battle of Agincourt occurred in what year? (I won’t ask this.)

1. Instead of splitting up a process’ address space into some number of variable-sized logical segments (e.g., code, heap, stack), we divide it into fixed-sized units, each of which we call a \_\_\_\_\_\_\_\_\_\_\_.
2. We’ll view physical memory as an array of fixed-sized units called \_\_\_\_\_\_\_\_\_\_\_\_\_\_; each of these page frames can contain a single virtual-memory page.
3. What is the major change advantage of paging over segmentation?
4. When using paging, translating a logical (virtual) address to a physical address, the high order bits represent \_\_\_\_\_\_\_\_\_\_\_\_\_.
5. When using paging, translating a logical (virtual) address to a physical address, the low order bits represent \_\_\_\_\_\_\_\_\_\_\_\_\_.
6. Given number of bits in bits for offset and virtual address, how many bits are left for virtual page number?

|  |  |  |  |
| --- | --- | --- | --- |
| **Page Size** | **Low Bits**  **(offset)** | **Virtual Address Bits** | **High Bits (vpn)** |
| 16 bytes | 4 | 10 |  |
| 1 KB | 10 | 20 |  |
| 512 bytes | 9 | 16 |  |
| 1 MB | 20 | 32 |  |
| 4 KB | 12 | 32 |  |

1. Given number of bits for virtual page number (vpn), how many virtual pages can there be in an address space?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Page Size** | **Low Bits**  **(offset)** | **Virtual Addr Bits** | **High Bits**  **(vpn)** | **Virtual Pages** |
| 16 bytes | 4 | 10 | 6 |  |
| 1 KB | 10 | 20 | 10 |  |
| 512 bytes | 9 | 16 | 5 |  |
| 1 MB | 20 | 32 | 12 |  |
| 4 KB | 12 | 32 | 20 |  |

1. Identify a couple advantages of paging.
2. Identify a couple disadvantages of paging.
3. Identify which of the following are cheap (fast) and which are expensive (slow).
   1. Extract VPN (virt page num) from VA (virt addr)
   2. Calculate addr of PTE (page table entry)
   3. Read PTE from memory
   4. Extract PFN (page frame num)
   5. Build PA (phys addr)
   6. Read contents of PA from memory into register
4. TLB stands for \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. When using a TLB, does the hit rate improve or worsen with smaller a page size?
6. What type of access pattern has a lower hit rate, sequential access or random access?
7. Describe special locality.
8. Describe temporal locality.
9. Which provides better performance, flushing the TLB with a context switch or tracking TLB entries with a process identifier?

**Week 5 material**

1. Threads are like processes, except \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. True or false: If one thread is blocked on I/O, the other threads in the process are also blocked.
3. Describe the difference between parallelism and concurrency.
4. Is it threads or processes where each has its own independent address space?
5. Define the term (in relation to our class): atomic.
6. Define the *term* (in relation to our class): critical region.
7. Define the *term* (in relation to our class): mutual exclusion.
8. Define the *term* (in relation to our class): race condition.
9. Define the term (in relation to our class): transaction.
10. True or false: Threads within a process share a single Page Table Base Register.
11. True or false: Threads within a process share a single Instruction Pointer.
12. True or false: Threads within a process share a single Stack Pointer.
13. Describe user level threads
14. Describe kernel level threads.
15. Mutex stands for \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
16. A mutex is always in 1 of 2 states. What are those states?
17. True or false: A thread can unlock a mutex that was locked by another thread.
18. True or false: A mutex that is unlocked can still be unlocked.
19. Describe a zombie thread.
20. Describe a detached thread.

**Week 6 material**

1. Under what conditions are spin locks fast?
2. Under what conditions are spin locks slow?
3. Describe deadlock (as used in our class).
4. The “deadly embrace” is another description for \_\_\_\_\_\_\_\_\_\_\_\_\_.
5. Describe the “Dining Philosophers Problem”.
6. What are the “Coffman Conditions”?
7. Elimination of any one of the Coffman Conditions will eliminate what?
8. Generally speaking, there are three ways of handling deadlocks. What are they?
9. What method do both Windows and Unix take for handling deadlock?

**Week 7 material**

1. True or false: There’s normally a brick wall between the memory of different processes; Process A cannot access or modify the memory of Process B. Shared memory allows you to change that.
2. True or false: Once a shared memory block is mapped into the address space of a process, it can be accessed exactly the same as rest of the address space for the process.
3. What means of synchronization or mutual-exclusion for multiple or concurrent access is provided by shared memory.
4. What C function is used to size a shared memory segment?
5. True or false: A POSIX shared memory segment will persist until explicitly removed or the system is rebooted.
6. For our class, a semaphore is an integer that cannot be \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
7. If the value of a semaphore is currently zero, then a request to decrement will \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
8. The initialization value for a semaphore represents \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
9. True or false: Semaphores can be used with either threads or processes.
10. True or false: Any process or thread can decrement a semaphore and any process or thread can increment a semaphore.
11. True or false: When a process exits, any semaphore it has locked will be automatically unlocked.
12. POSIX Semaphores come in 2 types, \_\_\_\_\_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
13. Describe the sem\_post() and sem\_wait() calls.

**Week 8 material**

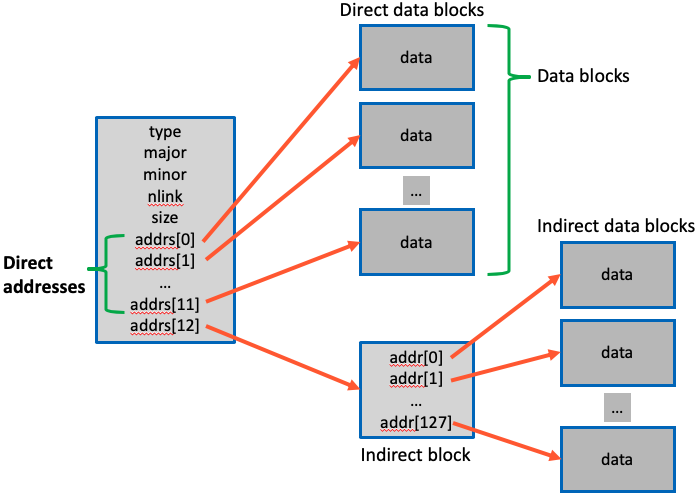
1. True or false: On Unix, a file can have more than one name.
2. Describe an inode for the Unix file system.
3. Write the C code to implement symbolic links in xv6. (Just kidding. ;-)
4. True or false: Like a file can have multiple names, it can have multiple inodes.
5. True or false: Both hard links and soft links (aka symbolic links) allow linking to directories.

**Week 9 material**

1. What is your favorite Thanksgiving tradition?

**Week 10 material**

1. Describe an inode.
2. Draw an example of “multi-level indexing” when allocating blocks for a file.
3. Draw the original block inode block allocation used in xv6.
4. Draw how we modified the inode block allocation for xv6 to support larger files.
5. What is fsck?
6. What is a super block?
7. Is a super block typically stored near the beginning or end of a disk drive?
8. What are some examples of the data stored in a super block?
9. The number of inodes on a disk represents the maximum number of \_\_\_\_\_\_\_\_\_\_.
10. A common way to keep track of free data blocks on a disk is to use a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
11. Using the diagram below and assuming that a disk block is 512 bytes, what is the maximum size of a file that can created?



1. What sort of disk block allocation and tracking method is used for the ext2, ext3, and ext4 file systems?