Full Name: ______________________

CS 450 Summer 2009
Midterm Exam

July 22, 2009

Instructions:

- This exam is open-book, open-notes.
- Keep your written answers concise and to-the-point. I reserve the right to deduct points for needless verbiage.
- Write your full name on the front, and make sure that your exam is not missing any sheets.
- Good luck!

Problem 1 (/10) :	
Problem 2 (/10) :	
Problem 3 (/5) :	
Problem 4 (/5) :	
Problem 5 (/5) :	
Problem 6 (/5) :	
Problem 7 (/8) :	
Problem 8 (/8) :	
Problem 9 (/9) :	
TOTAL (/65) :	

Problem 1. (10 points):

True/False. For each of the following statements, circle either `T` or `F` to indicate whether it is true or false.

`proc[0]`'s primary function is as the scheduler – i.e., it is responsible for the swapping of processes in and out of core.	`T` `F`
Each process in v6 (except for `proc[0]`) consists of kernel and user "halves". Switching execution between these two parts of a given process is accomplished through the use of the `savu` and `retu` functions.	`T` `F`
All interrupts in v6 are handled in kernel mode, but traps are generally handled in user mode so as to allow for system calls that result in process termination (and consequently never switch to kernel mode).	`T` `F`
The only processes in v6 that are eligible to be swapped out must be non-system processes and must also be blocked (e.g., waiting on an I/O request to complete).	`T` `F`
When an in-core process in v6 becomes blocked due to a pending I/O request, it is automatically scheduled to be swapped out to make room for more deserving processes that might reside in swap space.	`T` `F`

Problem 2. (10 points):

Multiple choice. For each of the following multiple choice problems, choose the *single best* answer by circling its corresponding letter.

1. Assume that the address of a process's user data area is stored in the top word of the stack. Which of the following lines of code effectively performs a context switch to this process??

   (a) `mov $_u, r1`

   (b) `mov (sp), KISA6`

   (c) `jsr pc, *(r0)+`

   (d) `mov $KISA0, r0`

2. Which of the following lines of code is to be executed if the scheduler is unable to find a process suitable for swapping out in order to make room for a runnable, swapped process?

   (a) `panic("swap error");`

   (b) `xwap(rp, 1, 0);`

   (c) `sleep(&runout, PSWP);`

   (d) `sleep(&runin, PSWP);`

3. Which of the following pieces of data associated with active processes always remains in core, regardless of the status of the process?

   (a) the prototype segmentation registers

   (b) the saved stack and frame pointers

   (c) the open file table

   (d) the process cpu usage

4. Which of the following lines of code in `swtch` performs a context switch to a newly selected user process?

   (a) `aretu(u.u_ssav)`

   (b) `retu(proc[0].p_addr)`

   (c) `savu(u.u_rsav)`

   (d) `retu(rp->p_addr)`

5. The current user struct can be accessed in the kernel via the pointer `_u`. Which of the following correctly initializes `_u`?

   (a) `_u = 140000`

   (b) `_u = *ka6`

   (c) `mov $USIZE-1\<8|6, _u`

   (d) `UISA->r[7] = ka6[1]`

Problem 3. (5 points):

Explain the purpose of the following lines of code towards the end of the `setrun` function (`rp` points to the argument to `setrun`):

```
if (runout != 0 && (rp->p_flag&SLOAD) == 0) {
    runout = 0;
    wakeup(&runout);
}
```

Problem 4. (5 points):

Where does the following section of code taken from `main` return to, and what is its purpose?

```
expand(USIZE+1);
estabur(0, 1, 0, 0);
copyout(icode, 0, sizeof icode);
return;
```

Problem 5. (5 points):

Explain the function of the following code. When might you expect to find it executed?

```
2:
  bis  $340, PS
  tstb _runrun
  beq  2f
  bic  $340, PS
  jsr  ps,_swtch
  br   2b
2:
```

Problem 6. (5 points):

In the clock interrupt handler function, just before invoking any scheduled callout functions, we find the following section of code:

```
if ((ps&0340) != 0)
    goto out;
```

What is the purpose of this code?

Problem 7. (8 points):

Consider the following parameters of an i-node based, UFS-like file system:

- Blocks are 512 bytes large
- Block pointers are 64-bits (8 bytes) wide
- Each i-node consists of 8 direct pointers, 2 single indirect pointers, and 1 double indirect pointer

1. What is the largest file size supported by this filesystem? (Show your work)

2. Assume that we already have the i-node for a given file in memory, and that a read request arrives for byte 115,200 of that file. How many disk reads (i.e., individual block requests) must be performed to satisfy that request? Explain.

Problem 8. (8 points):

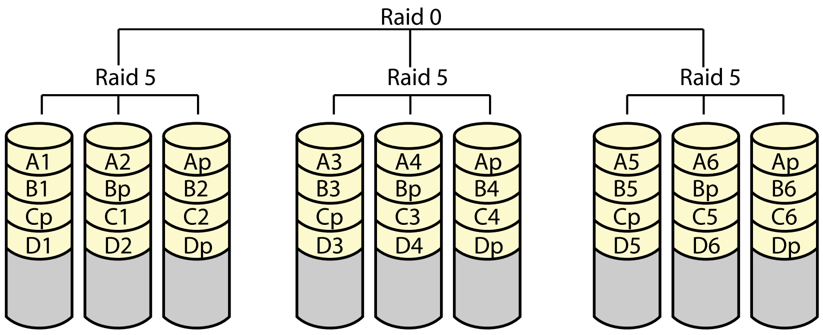
Durability (one of the ACID properties) of write operations is not generally enforced in the UNIX I/O architecture.

1. Justify this design decision. What would be the cost of guaranteeing durability of all write operations?

2. Explain how journaling helps to ensure filesystem consistency in spite of the lack of durable writes.

Problem 9. (9 points):

RAID 5+0 is a form of nested RAID where a single RAID 0 array is striped over multiple RAID 5 sets. The following diagram illustrates how RAID 5+0 would be implemented using 9 drives organized into 3 separate RAID 5 sets (each set using distributed parity).



1. What is the maximum number of drives that can fail in the example above without irrecoverable data loss? Explain.

2. What is the data storage capacity of the RAID 5+0 setup above? (Relative to the total disk space, assuming all drives are the same size).

3. How does RAID 5+0 improve the write performance of RAID 5?