CS 4061, Spring 2000, Exam 1

Name:_______________________ ID#___________________________

Q1. Abstraction. (20 pts.)

We said that an operating system should provide both vertical abstraction (within a machine) and horizontal abstraction (across machines). Give one advantage of each abstraction type and explain, in general terms, how the O/S can provide each advantage.

A. Vertical abstraction.

B. Horizontal abstraction.

Q2. Resource Management for Processes. (20 pts.)

A. Temporal multiplexing.

We said that in a time-sharing system processes are given "time slices" of access to the CPU. We also defined a process to be a program in execution and said that a process could be in one of several states.

When a time slice expires but a process is not done executing, that process moves from what execution state to what other execution state?

Why is this done (rather than just letting the process run to completion)?

Name two pieces of information about the process that need to be saved when this transition occurs and explain why they need to be saved.

B. Protection.

How can an O/S protect one process from interference from another process (such as having its memory overwritten)?

What hardware is needed to enforce this protection? **Why?**

Q3. Process Control. (20 pts.)

```
% man -k convert | grep pnm | grep ps
```

A. `fork` and `exec`.

For a shell to execute the given command line, how many times would it `fork`? ____
How many time would it call an `exec` family function? ____
Explain your answers.

B. Process termination.

If the shell contained the following code, executed before it begins to read commands from the prompt, how many times would the `printf` statement be called when executing the command line given above? (Assume that the registration of the exit function succeeds.)
Explain your answer.

```
...
void my_exit_fcn (void){
   printf ("Exiting shell ... \n");
}
...
atexit(my_exit_fcn);
...
/* code for reading from command lines, forking, exec-ing,
etc. */
...
```

Q4. Process Control, again. (20 pts.)

```
...
int main(void){
   pid_t cpid;   int status;
   cpid = fork();
   if (cpid == 0) {
      printf("Child Process\n");
      if (wait (NULL) > 0)
         exit(0);
      else
         exit(1);
   } else if (cpid > 0){
      waitpid(-1, &status, WNOHANG);
      printf("Parent Process\n");
      exit(0);
   } else {
      printf("Fork failed\n");
      exit(2);
   }
}
```

Given the code above, which statement will be printed first on a POSIX-compliant system, assuming that the `fork` succeeds? **Explain your answer.**

What value will be returned by the parent process? **Explain your answer.**

What value will be returned by the child process? **Explain your answer.**

If this program is compiled and run from a shell, what value will be returned to the shell? **Explain your answer.**

Q5. Signals. (20 pts.)

```
/* Code Style 1 */
...
/* fork off child */
...
while (cpid = waitpid(-1, &status, WNOHANG) == 0)
   sleep(30);
...
/* deal with exited child */
...
```

```
/* Code Style 2 */
...
void handle_child_exit(int);
static struct sigaction act;
act.sa_handler = handle_child_exit;
sigfillset(&(act.sa_mask));
sigaction(SIGCHLD, &act, NULL);
...
/* fork off child */
...
void handle_child_exit(int signo){
  ...
   /* deal with exited child */
  ...
}
```

List and explain two advantages of the second coding style over the first.