CS 390
Chapter 3 Homework Solutions

3.1 `Using the program shown in ...`

```
PARENT: value = 5
```

3.2 `Including the initial parent process, ...`

8

3.4 `Some computer systems provide multiple ...`

When a context switch occurs, we must load the saved context of the kernel (if we are switching from user to kernel mode) or the next scheduled process (if we are switching from kernel mode to user mode) into the CPU registers.

If the new context is already loaded into one of the register sets, we simply adjust the "window pointer" (a CPU register available only to the kernel) to point to the correct register set.

If the new context is in memory (in the process's PCB), but there is a free register set, the kernel loads the new process's context into a register set, and adjusts the window pointer to point to the new context.

If the new context is in memory, and all the register sets are in use, the kernel must choose a process and "evict" that process's context from the register set by writing it to the evicted process's PCB. Once the write is complete, the kernel loads the new context into the newly available register set and adjusts the window pointer.

3.5 `When a process creates a ...` Remember that a `fork()` creates a *copy* of the process, meaning that the process's entire address space, along with its PCB, is copied.

a. Not shared.

b. Not shared.

c. Shared. To understand why, realize that shared memory segments are outside of the process's address space, and that the processs accesses them through pointers. The pointers themselves *are* in the processes address space.

Additional Exercises: Answers to the additional exercises have been moved to the next page.

A Describe the actions taken by the kernel when a context switch occurs.

B List two different specific events that can cause a context switch.

A Solution Assume that the context switch is caused by the process making a system call. (Device interrupts are handled similarly.)

1. The mode bit is set to `kernel`. (This happens in hardware, and so is not a precisely a kernel action.)
2. The context of the currently executing process is saved in its PCB.
3. The context of the kernel is loaded.
4. The timer chip is disabled.
5. The kernel checks to see what type of interrupt was signalled. In this case, it was a trap from a user process, so it executes the "syscall service" routine.
6. In the syscall service routine, the kernel checks the legality of the parameters passed with the syscall. For example, if the kernel is passed any pointers, the pointers should point to memory in the process's address space. If some naughty process tried to pass a bad pointer to the kernel, the kernel will terminate the process.
7. The kernel performs the syscall or (for IO) starts the data transfer to/from the appropriate device.
8. The kernel moves the process to the wait queue (if the process needs to wait on a device), the ready queue (if the process can execute immediately), or the terminating queue (if the process tried to perform an illegal action or called `exit()`.
9. The kernel runs the short-term scheduler to select another process P_j to execute.
10. The timer chip is re-enabled.
11. The kernel's context is saved.
12. The context of P_j is loaded.
13. The kernel sets the mode bit to user mode.
14. The kernel loads the next instruction of P_j into the CPU's instruction pointer.

B solution Here are four.

1. On a multitasking system, expiration of a process's time-slice (indicated by an interrupt from the timer) causes a context switch to the kernel.
2. A request to perform a system call causes a context switch to the kernel.
3. The arrival of an interrupt from an external device causes a context switch to the kernel. Examples: A disk sends an interrupt when it has completed a previously requested DMA operation. A network card sends an interrupt when a packet arrives. The keyboard sends an interrupt when the user presses a key.

4. When the kernel finishes processing a system call or servicing an interrupt, it performs a context switch from the kernel back to a user process.