System Call Sheet

Lab Exercises:

1. Write a C program to block a parent process until child completes using wait system call.
2. Write a C program to load the binary executable of the previous program in a child process using exec system call.
3. Write a program to create a child process. Display the process IDs of the process, parent and child (if any) in both the parent and child processes.
4. Create a zombie (defunct) child process (a child with exit() call , but no corresponding wait() in the sleeping parent) and allow the init process to adopt it (after parent terminates). Run the process as background process and run "ps" command.
5. Write a C program to create a file and write contents to it.
6. Write a C program to copy the content of one file to other.

Additional Exercises:

1. Create a orphan process (parent dies before child – adopted by "init" process) and display the PID of parent of child before and after it becomes orphan. Use sleep(n) in the child to delay the termination.
2. Modify the program in the previous question to include wait(&status) in the parent and to display the exit return code(left most byte of status) of the child.
3. Use lseek() to copy different parts(initial, middle and last) of the file to other. (For lseek() refer to man pages)
4. Create a child process which returns a 0 exit status when the minute of time is odd and returning a non-zero (can be 1) status when the minute of time is even.