IIT Bhubaneswar
School of Electrical Sciences
Operating Systems Lab (0-0-3)
Spring 2017
Lab Schedule: Thu (8:00-10:55)

Instructor: Debi Prosad Dogra (dpdogra@iitbbs.ac.in)
Teaching Assistant: Santosh K. K. (sk47@iitbbs.ac.in)

Laboratory 2 (**Submission Deadline: 27/01/2017 midnight**)

Points: **20 (including practice)**

Objectives: The goal of this lab is to give concepts on process creation, execution, and multiprogramming.

Practice during Laboratory Hours:

- Understanding the following systems calls related to process creation, execution, and termination: fork(), wait(), waitpid(), sleep(), getpid(), getppid(), execl(), execlp(), execv(), execvp(), execve(), abort(), system(), setuid(), setgid(), gettimeofday(), settimeofday().
- Write small code fragments to check the utility of each of the above system calls and see the outputs. Carefully read the man pages for the above mentioned system calls to understand various arguments and settings.
- pid_t wait(int *status) is a call to wait() that returns the pid of a terminated child or -1 on error. If no child has terminated, the call blocks until a child terminates. If a child has already terminated, the call returns immediately. Consequently, a call to wait( ) in response to news of a child's demise–say, upon receipt of a SIGCHLD-will always return without blocking. On error, there are two possible errno values: ECHILD-The calling process does not have any children. EINTR-A signal was received while waiting, and the call returned early. If not NULL, the status pointer contains additional information about the child. Because POSIX allows implementations to define the bits in status as they see fit, the standard provides a family of macros for interpreting the parameter:

```
#include <sys/wait.h>
int WIFEXITED (status);
int WIFSIGNALED (status);
int WIFSTOPPED (status);
int WIFCONTINUED (status);
int WEXITSTATUS (status);
int WTERMSIG (status);
int WSTOPSIG (status);
int WCOREDUMP (status);
```

Now, see the code listed below and try to understand what happens when it gets executed:

```
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
```

```
int main (void)
{
    int status;
    pid_t pid;
    if (!fork ( ))
            return 1;
    pid = wait (&status);
    if (pid == -1)
            perror ("wait");
    printf ("pid=%d\n", pid);
    if (WIFEXITED (status))
            printf ("Normal termination with exit status=%d\n", WEXITSTATUS (status));
    if (WIFSIGNALED (status))
            printf ("Killed by signal=%d%s\n", WTERMSIG (status), WCOREDUMP (status)
            ? " (dumped core)" : "");
    if (WIFSTOPPED (status))
            printf ("Stopped by signal=%d\n", WSTOPSIG (status));
    if (WIFCONTINUED (status))
            printf ("Continued\n");
    return 0;
}
```

If, instead of having the child return, it calls abort(), which sends itself the SIGABRT signal, something different will be printed. Examine that through sample program.

Take Home Assignment for Submission:

Will be given during the laboratory hour.

Submission Guidelines: One PDF file containing answers for each part of the assignment. Explain in your own words. Copying will affect your grade heavily. Electronic submission has to be sent to the TA.

Additional Reading: Read through **Chapter 1** in the **xv6** textbook.