UNIVERSITY AT ALBANY
State University of New York

COLLEGE OF ENGINEERING AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

ICSI333 System Fundamentals

Note: Students are expected to start the activities as soon as the description is available and seek feedback as needed. Although some activities are not graded for credit, they are contiguous study of the lecture or used as stepping-stones for the projects. Skipping any activities would impact the learning significantly.

Objectives:

- Practices on processes and pipes

Reading:

- Lecture notes 14 and 15
- Reading materials

Submission (5 points):

- All required C programs must be submitted on Duifene on time.

Instructions:

Note: If an example generates warnings or errors on your system, fix them first.

Task #1. with a partner, study and discuss the following examples. You may compile and run the examples if needed.

Example 1:

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(){
      fork();
      printf("Hello world!\n");
      return 0;
}
```

- What is the output of this program?

Example 2:

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

int main(void){
      pid_t pid;
      int rv;

      switch(pid = fork()) {
      case -1:
            perror("fork"); /* something went wrong */
            exit(1); /* parent exits */
      case 0:
            printf(" CHILD: This is the child process!\n");
            printf(" CHILD: My PID is %d\n", getpid());
            printf(" CHILD: My parent's PID is %d\n", getppid());
            exit(0);
      default:
            printf("PARENT: This is the parent process!\n");
```

```
                printf("PARENT: My PID is %d\n", getpid());
                printf("PARENT: My child's PID is %d\n", pid);
          }
          return 0;

     }
```

- What are possible return values from `fork()` ?
- What does it mean if `fork()` returns 0?
- What function is used to get a parent's PID?

Example 3:

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

int main(void){
   pid_t pid;
   int rv;

   switch(pid = fork()) {
     case -1:
         perror("fork"); /* something went wrong */
         exit(1); /* parent exits */
     case 0:
         printf(" CHILD: This is the child process!\n");
         printf(" CHILD: My PID is %d\n", getpid());
         printf(" CHILD: My parent's PID is %d\n", getppid());
         printf(" CHILD: Enter my exit status (make it small): ");
         scanf(" %d", &rv);
         printf(" CHILD: I'm outta here!\n");
         exit(rv);
     default:
         printf("PARENT: This is the parent process!\n");
         printf("PARENT: My PID is %d\n", getpid());
         printf("PARENT: My child's PID is %d\n", pid);
         printf("PARENT: I'm now waiting for my child to exit()...\n");
         wait(&rv);
         printf("PARENT: My child's exit status is: %d\n", WEXITSTATUS(rv));
         printf("PARENT: I'm outta here!\n");
     }

     return 0;
  }
}
```

- What is the purpose of function `wait()`?
- What are the parameters of function `wait()`?
- What does `WEXITSTATUS()` return?

Example 4:

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

int main(void) {
   pid_t child;
```

```
    int cstatus; /* Exit status of child. */
    pid_t c; /* Pid of child to be returned by wait. */

    if ((child = fork()) == 0) {
          /* Child process. To begin with, it prints its pid. */
          printf("Child: PID of Child = %ld\n", (long) getpid());

          /* Wait for an input*/
          printf("Press Enter to continue\n");
          getchar();

          /* Child will now execute the nano command. */
          execlp("nano", "nano", "man2fork.txt", NULL);

          fprintf(stderr, "Child process could not do execlp.\n");
          exit(1);
    }else { /* Parent process. */
          if (child == (pid_t)(-1)) {
                fprintf(stderr, "Fork failed.\n"); exit(1);
          }else {
                c = wait(&cstatus); /* Wait for child to complete. */
                printf("Parent: Child %ld exited with status = %d\n", c,
                WEXITSTATUS(cstatus));
          }
    }

    return 0;
 }
```

- Does the program execute the highlighted `fprintf` statement?
- What is the difference between `fork()` and `exec()`?

Example 5:

```
#include<stdio.h>

int main(){
    while(1){
          printf("hello world\n");
          sleep(1);
    }
    return 0;
}
```

- Compile and run the program, and press `Ctrl-C`. What happened? Which signal was sent to the process?

Example 6:

```
    #include<stdio.h>
    #include<signal.h>

    void handle_sigint(int sig){
          printf("Caught signal %d\n", sig);
    }

    int main(){
          int flag = 5;
          signal(SIGINT, handle_sigint);

          while(flag){
                flag--;
          sleep(10);
```

```
        }
        return 0;
}
```

- Compile and run the program, and press `Ctrl-C`. What happened? Why is it different from the previous program?

Task #2. Write a computer program that provides the following functionalities.
Create two C programs: `p1.c` and `p2.c.`

- `p1` calls `fork` to create a child process. The child process must execute `p2,` a difference program.
- `p1` and `p2` must be able to send signals (SIGUSR1, SIGUSR2) to themselves and to each other. The signals must be caught and handled by the respective process properly.
  - In the signal handler, the pid of the process and the signal information must be printed.
- `p1` and `p2` must be able to transfer data via a FIFO.
  - `p1` writes to a FIFO. Any characters can be written to the FIFO by `p1`.
  - `p2` reads from the same FIFO. If a byte to be read is a digit, prints 'N'; if a byte to be read is a letter, prints 'L'; for otherwise, prints 'O'.

Compile both programs and name each executable program. Run the executable of `p1`, verify the results.

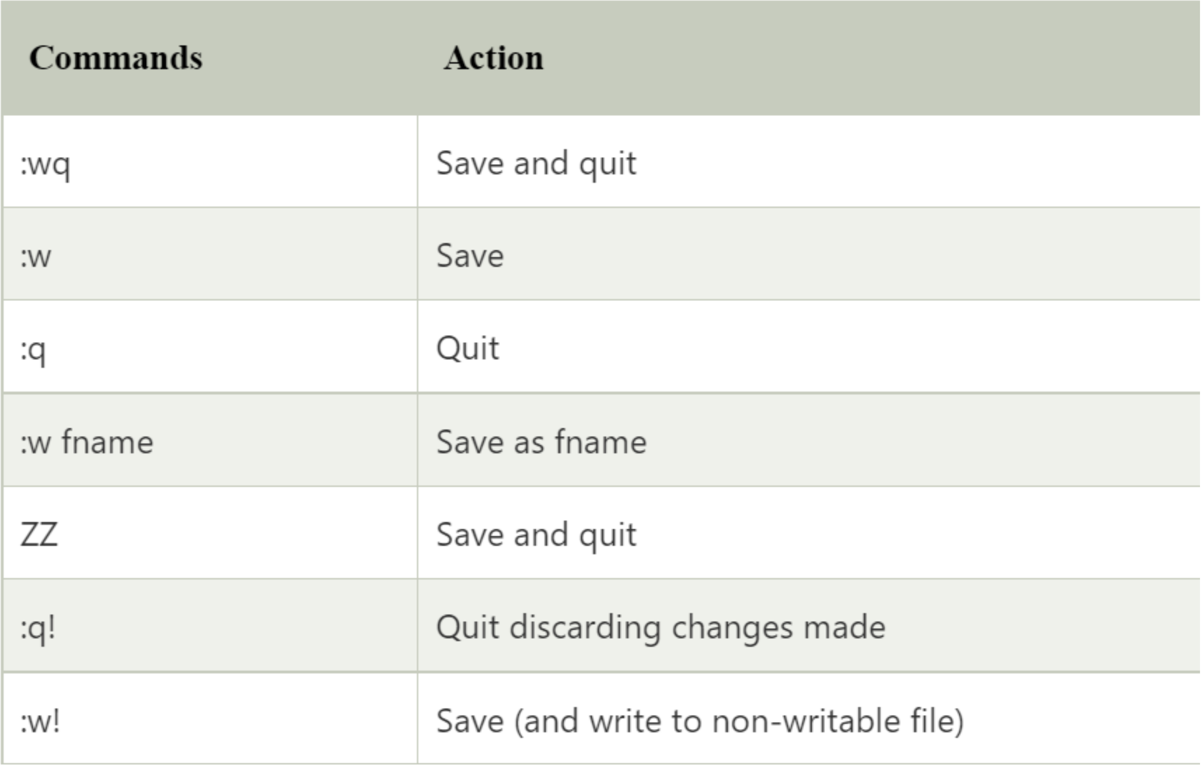
Linux Basic Commands

`ls [option(s)] [file(s)]`	If you run ls without any additional parameters, the program will list the contents of the current directory in short form. `-l` detailed list `-a` displays hidden files
`cp [option(s)] sourcefile targetfile`	Copies sourcefile to targetfile. `-I` waits for confirmation, if necessary, before an existing targetfile is overwritten `-r` copies recursively (includes subdirectories)
`mv [option(s)] sourcefile targetfile`	Copies sourcefile to targetfile then deletes the original sourcefile. `-b` creates a backup copy of the sourcefile before moving. `-I` waits for confirmation, if necessary, before an existing targetfile is overwritten.
`rm [option(s)] file(s)`	Removes the specified files from the file system. Directories are not removed by rm unless the option `-r` is used. `-r` deletes any existing subdirectories `-I` waits for confirmation before deleting each file
`cd [options(s)] [directory]`	Changes the current directory. cd without any parameters changes to the user's home directory.
`mkdir [option(s)] directoryname`	Creates a new directory.
`rmdir [option(s)] directoryname`	Deletes the specified directory, provided it is already empty.
`cat [option(s)] file(s)`	The cat command displays the contents of a file, printing the entire contents to the screen without interruption. `-n` numbers the output on the left margin
`cal`	Displays the calendar of the current month.
`date`	Displays current time and date.
`whoami`	Reveals the user who is currently logged in.
`whatis`	Gives a one-line description about the command. It can be used as a quick reference for any command.

`man`	Manual Pages, for more detailed information, Linux provides man pages and info pages. To see a command's manual page, man command is used.
`pwd`	Prints the absolute path to current working directory.
`vi`	A text editor for Linux operating system.
`gedit`	The default GUI text editor in the Ubuntu operating system.
`emacs`	Another text editor for Linux operating system.

Vi commands:

To save and quit:



Commands	Action
:wq	Save and quit
:w	Save
:q	Quit
:w fname	Save as fname
ZZ	Save and quit
:q!	Quit discarding changes made
:w!	Save (and write to non-writable file)

More commands:

Copy-pasting within the terminal: Ctrl+Shift+C/V

Resources and Credits:

- Teaching materials from Professor Kuperman at UAlbany
- Vi reference: https://www.ele.uri.edu/faculty/vetter/Other-stuff/vi/vi-quick-ref.pdf