**CISC2005 Exercise 03**

# Prerequisites

## Successful installation of \*nix terminal (e.g., WSL, cygwin) or a \*nix operating system (e.g., Linux, Mac).

For Windows system, I recommend you to install WSL. <https://learn.microsoft.com/en-us/windows/wsl/install>

## Successful installation of a C compiler, GCC is recommended.

## Successful installation of a text editor, VIM or Emacs is recommended.

# Tasks

In this section, students are required to execute the following codes, and try to understand the behaviour. For each program, students should capture a screenshot of the successful execution, and answer attached questions briefly. In the submission file, please attach the **execution screenshot** and the **explanation of each question** in sequence.

Q1: Execute the code several times, and explain why the variable *pid* is consistent/inconsistent.

*#include <stdio.h>*

*#include <sys/types.h>*

*#include <unistd.h>*

*int main() {*

*pid\_t pid;*

*int i = 5;*

*printf("Hello EveryOne :), the value of i is %d: This statement before fork()\n", i);*

*pid = fork();*

*printf("The value of pid is %d. Thsi statement after fork(), Goodbye :)\n", pid);*

*return 0;*

*}*

Q2: please explain the result of the variable *myVariable* in output.

*#include <stdio.h>*

*#include <stdlib.h>*

*#include <unistd.h>*

*int main(int argc, char \*argv[]) {*

*int myVariable = 100;*

*printf("hello world (pid:%d)\n", (int)getpid());*

*printf("Value of myVariable before fork is %d\n", myVariable);*

*int rc = fork();*

*if (rc < 0) {*

*// fork failed; exit`*

*fprintf(stderr, "fork failed\n");*

*exit(1);*

*}*

*else if (rc == 0) {*

*myVariable = 200;*

*printf("The changed value of myVariable from child is %d\n", myVariable);*

*} else {*

*myVariable = 300;*

*printf("The changed value of myVariable from parent is %d\n", myVariable);*

*}*

*return 0;*

*}*

Q3: Please explain the frequency of each level printed in the output.

*#include <stdio.h>*

*#include <sys/types.h>*

*#include <unistd.h>*

*int main(){*

*pid\_t pid1, pid2, pid3;*

*pid1 = 0; pid2 = 0; pid3 = 0;*

*pid1 = fork();*

*if (pid1 == 0) {*

*pid2 = fork();*

*pid3 = fork();*

*} else {*

*pid3 = fork();*

*if (pid3 == 0)*

*pid2 = fork();*

*}*

*if ((pid1 == 0) && (pid2 == 0))*

*printf("level1\n");*

*if (pid1 != 0)*

*printf("level2\n");*

*if (pid2 != 0)*

*printf("level3\n");*

*if (pid3 != 0)*

*printf("level4\n");*

*return 0;*

*}*

Q4: Please explain that if the *wait()* function can be replaced by waiting for a while using a for-loop? Why?

*#include <stdio.h>*

*#include <sys/types.h>*

*#include <sys/wait.h>*

*int main() {*

*int status;*

*pid\_t pid;*

*pid = fork();*

*if (pid == -1)*

*printf("\n ERROR child not created.");*

*else if (pid == 0) {*

*printf("\n I'm the child!");*

*exit(0);*

*} else {*

*pid\_t TCpid;*

*TCpid = wait(&status); //wait returns a exiting status of child process, as well as the child’s pid*

*printf("\n I'm the parent!");*

*printf("\n The child with pid = %d terminated with a status = %d \n", TCpid, status);*

*}*

*return 0;*

*}*

Q5: Is it possible the “Child Complete” printed before execution of subprocess? What does wait(NULL) exactly mean?

*#include <stdio.h>*

*#include <sys/types.h>*

*#include <unistd.h>*

*#include <stdlib.h>*

*int main() {*

*pid\_t pid;*

*pid = fork();*

*if (pid < 0) {*

*fprintf(stderr, "Fork Failed");*

*exit(-1);*

*} else if (pid == 0) {*

*char \*\* args;*

*args = malloc(3 \* sizeof(char\*));*

*args[0] = "ls";*

*args[1] = "-l";*

*execv("/bin/ls", args);*

*} else {*

*wait(NULL);*

*printf("Child Complete\n");*

*exit(0);*

*}*

*}*

Q6: Please try three different commands using *myShell* and capture the screenshot of execution. (e.g., ls, pwd.)

*#include <stdio.h>*

*#include <string.h>*

*#include <unistd.h>*

*#include <stdlib.h>*

*int count(char\* buffer) {*

*int count = 0;*

*char\* argument;*

*argument = strtok(buffer, " \n");*

*while(argument != NULL) {*

*count++;*

*argument = strtok(NULL, " \n");*

*}*

*return count;*

*}*

*int main() {*

*char buffer[512];*

*char\* path = "/bin/";*

*while(1) {*

*printf("myShell>");*

*fgets(buffer, 512, stdin);*

*int pid = fork();*

*if (pid != 0) {*

*wait(NULL);*

*} else {*

*int no\_of\_args = count(buffer);*

*char\*\* array\_of\_strings = malloc((sizeof(char\*)\* (no\_of\_args+1)));*

*int i = 0;*

*char\* ptr;*

*ptr = strtok(buffer, " \n");*

*while(ptr != NULL) {*

*array\_of\_strings[i] = (char\*)malloc((sizeof(char)\*strlen(ptr)));*

*strcpy(array\_of\_strings[i], ptr);*

*ptr = strtok(NULL, " \n");*

*i++;*

*}*

*char\* prog = malloc((sizeof(char)\*(no\_of\_args+1)));*

*prog = strcat(strcpy(prog, path), array\_of\_strings[0]);*

*int rv = execv(prog, array\_of\_strings);*

*}*

*}*

*return 0;*

*}*

Q7: After executing the following code, a new file named *myFile.txt* is generated. Is the content in *myFile.txt* will be consistent? Why?

*#include <stdio.h>*

*#include <stdlib.h>*

*#include <unistd.h>*

*#include <fcntl.h>*

*#include <errno.h>*

*#include <sys/wait.h>*

*int main(int argc, char \*argv[]){*

*printf("hello world (pid:%d)\n", (int)getpid());*

*int fd = open("myFile.txt", O\_CREAT|O\_RDWR);*

*if(fd == -1 ) {*

*printf("Unable to open the file\n exiting....\n");*

*return 0;*

*}*

*int rc = fork();*

*if (rc < 0) {*

*fprintf(stderr, "fork failed\n");*

*exit(1);*

*}*

*else if (rc == 0) {*

*printf("hello, I am child (pid:%d)\n", (int)getpid());*

*char myChar='a';*

*write(fd, &myChar,1);*

*printf("writing a character to the file from child\n");*

*}*

*else {*

*printf("hello, I am parent of %d (pid:%d)\n",*

*rc, (int)getpid());*

*char myChar='b';*

*write(fd, &myChar,1);*

*printf("writing a character to the file from parent\n");*

*}*

*return 0;*

*}*