Process Management

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <errno.h>

int **main**() {

    pid\_t child\_pid;

    int status;

*// Fork a child process*

    child\_pid = **fork**();

    if (child\_pid < 0) {

**perror**("fork failed");

**exit**(EXIT\_FAILURE);

    } else if (child\_pid == 0) {

*// This is the child process*

**printf**("Child process: PID = %d\n", **getpid**());

**printf**("Child process: PPID = %d\n", **getppid**());

**printf**("Child process: Executing ls command...\n");

**execlp**("ls", "ls", "-l", NULL);

*// execlp() will only return if there's an error*

**perror**("execlp failed");

**exit**(EXIT\_FAILURE);

    } else {

*// This is the parent process*

**printf**("Parent process: PID = %d\n", **getpid**());

**printf**("Parent process: Waiting for child process to terminate...\n");

**wait**(&status);

        if (**WIFEXITED**(status)) {

**printf**("Parent process: Child process exited with status %d\n", **WEXITSTATUS**(status));

        } else {

**printf**("Parent process: Child process exited abnormally\n");

        }

    }

    return 0;

}

Information Maintenance

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

int **main**() {

*// Get and print the process ID (PID) of the current process*

    pid\_t pid = **getpid**();

**printf**("Process ID (PID): %d\n", pid);

*// Get and print the parent process ID (PPID) of the current process*

    pid\_t ppid = **getppid**();

**printf**("Parent Process ID (PPID): %d\n", ppid);

*// Get and print the user ID (UID) of the current user executing the program*

    uid\_t uid = **getuid**();

**printf**("User ID (UID): %d\n", uid);

    return 0;

}

File Management

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

int **main**() {

    int fd1, fd2;

    ssize\_t bytes\_read, bytes\_written;

    char buffer[1024];

*// Open input file "input.txt" for reading*

    fd1 = **open**("input.txt", O\_RDONLY);

    if (fd1 == -1) {

**perror**("Failed to open input file");

**exit**(EXIT\_FAILURE);

    }

*// Open output file "output.txt" for writing (create if not exists, truncate if exists)*

    fd2 = **open**("output.txt", O\_WRONLY | O\_CREAT | O\_TRUNC, S\_IRUSR | S\_IWUSR);

    if (fd2 == -1) {

**perror**("Failed to open output file");

**exit**(EXIT\_FAILURE);

    }

*// Read from input file and write to output file*

    while ((bytes\_read = **read**(fd1, buffer, sizeof(buffer))) > 0) {

        bytes\_written = **write**(fd2, buffer, bytes\_read);

        if (bytes\_written != bytes\_read) {

**perror**("Write error");

**exit**(EXIT\_FAILURE);

        }

    }

    if (bytes\_read == -1) {

**perror**("Read error");

**exit**(EXIT\_FAILURE);

    }

*// Close input and output files*

    if (**close**(fd1) == -1) {

**perror**("Failed to close input file");

**exit**(EXIT\_FAILURE);

    }

    if (**close**(fd2) == -1) {

**perror**("Failed to close output file");

**exit**(EXIT\_FAILURE);

    }

**printf**("File copied successfully.\n");

    return 0;

}