컴퓨터학부 20201841 박세연

<execv1>

1. 소스코드

**[ssu\_execv\_1.c]**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <sys/resource.h>

#include <sys/wait.h>

// 함수 원형 선언

double ssu\_maketime(struct timeval \*time);

void term\_stat(int stat);

void ssu\_print\_child\_info(int stat, struct rusage \*rusage);

int main(void) {

struct rusage rusage; // 자원 사용량 정보를 저장할 구조체

pid\_t pid; // 자식 프로세스 ID 저장 변수

int status; // 자식 프로세스의 종료 상태를 저장할 변수

// 자식 프로세스 생성

if ((pid = fork()) == 0) { // 자식 프로세스인 경우

char \*args[] = {"find", "/", "-maxdepth", "4", "-name", "stdio.h", NULL}; // execv로 실행할 명령어 및 옵션

if (execv("/usr/bin/find", args) < 0) { // execv로 명령어 실행

fprintf(stderr, "execv error\n"); // 오류 발생 시 에러 메시지 출력

exit(1); // 프로세스 종료

}

}

// 부모 프로세스가 자식 프로세스의 종료를 대기

if (wait3(&status, 0, &rusage) == pid) // 자식 프로세스의 종료를 기다림

ssu\_print\_child\_info(status, &rusage); // 종료 정보 출력

else {

fprintf(stderr, "wait3 error\n"); // wait3 함수 실패 시 에러 메시지 출력

exit(1); // 프로세스 종료

}

exit(0); // 프로그램 종료

}

// timeval 구조체를 이용하여 시간을 계산하는 함수

double ssu\_maketime(struct timeval \*time) {

return ((double) time->tv\_sec + (double) time->tv\_usec / 1000000.0); // 초 단위 시간 반환

}

// 자식 프로세스의 종료 상태에 대한 정보를 출력하는 함수

void term\_stat(int stat) {

if (WIFEXITED(stat)) // 자식 프로세스가 정상적으로 종료된 경우

printf("normally terminated. exit status = %d\n", WEXITSTATUS(stat)); // 종료 상태 출력

else if (WIFSIGNALED(stat)) // 자식 프로세스가 시그널에 의해 종료된 경우

printf("abnormal termination by signal %d. %s\n", WTERMSIG(stat),

#ifdef WCOREDUMP

WCOREDUMP(stat) ? "core dumped" : "no core" // 코어 덤프 여부 출력

#else

NULL

#endif

);

else if (WIFSTOPPED(stat)) // 자식 프로세스가 중단된 경우

printf("stopped by signal %d\n", WSTOPSIG(stat)); // 중단된 시그널 출력

}

// 자식 프로세스의 종료 정보와 자원 사용량 정보를 출력하는 함수

void ssu\_print\_child\_info(int stat, struct rusage \*rusage) {

printf("Termination info follows\n"); // 종료 정보 출력 시작

term\_stat(stat); // 종료 상태 정보 출력

printf("user CPU time : %.2f(sec)\n", ssu\_maketime(&rusage->ru\_utime)); // 사용자 CPU 시간 출력

printf("system CPU time : %.2f(sec)\n", ssu\_maketime(&rusage->ru\_stime)); // 시스템 CPU 시간 출력

}

1. 실행결과

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

<execv2>

1. 소스코드

**[ssu\_execv\_2.c]**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main(void) {

char \*argv[] = {

"ssu\_execl\_test\_1", "param1", "param2", (char \*)0 // execv에 전달할 실행 파일 이름과 인자 목록

};

printf("this is the original program\n"); // 프로그램 시작을 알리는 메시지 출력

execv("./ssu\_execl\_test\_1", argv); // ssu\_execl\_test\_1 프로그램을 새로운 프로세스로 실행

printf("%s\n", "This line should never get printed\n"); // execv가 성공적으로 실행되었을 경우 이 코드는 실행되지 않음

exit(0); // 프로그램 종료

}

2. 실행결과

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<execl1>

1. 소스코드

**[ssu\_execl\_test\_1.c]**

#include <stdio.h>

#include <stdlib.h>

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

extern char \*\*environ; // 전역 변수로 선언된 환경 변수 배열

char \*\*str; // 환경 변수 배열을 탐색하기 위한 포인터

int i;

// 전달된 명령행 인자들을 출력하는 반복문

for (i = 0; i < argc; i++)

printf("argv[%d]: %s\n", i, argv[i]); // argv 배열의 각 요소를 출력

// 환경 변수 배열을 탐색하면서 각 환경 변수의 값을 출력하는 반복문

for (str = environ; \*str != 0; str++)

printf("%s\n", \*str); // 환경 변수의 값을 출력

exit(0); // 프로그램 종료

}

**[ssu\_execl\_1.c]**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main(void) {

printf("this is the original program\n"); // 프로그램 시작을 알리는 메시지 출력

execl("./ssu\_execl\_test\_1", "ssu\_execl\_test\_1", "param1", "param2", "param3", (char \*)0);

// ssu\_execl\_test\_1 프로그램을 실행하고자 함. 실행 파일 이름과 전달할 인자들을 지정

printf("%s\n", "this line should never get printed\n"); // execl이 성공적으로 실행되면 이 코드는 실행되지 않음

exit(0); // 프로그램 종료

}

2. 실행결과

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<execl2>

1. 소스코드

**[ssu\_execl\_2.c]**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main(void) {

// 첫 번째 자식 프로세스 생성

if (fork() == 0) { // 자식 프로세스인 경우

execl("/bin/echo", "echo", "this is", "message one", (char \*)0); // echo 명령어를 실행하여 "this is message one" 출력

fprintf(stderr, "exec error\n"); // execl 함수가 실패한 경우 에러 메시지 출력

exit(1); // 자식 프로세스 종료

}

// 두 번째 자식 프로세스 생성

if (fork() == 0) { // 자식 프로세스인 경우

execl("/bin/echo", "echo", "this is", "message two", (char \*)0); // echo 명령어를 실행하여 "this is message two" 출력

fprintf(stderr, "exec error\n"); // execl 함수가 실패한 경우 에러 메시지 출력

exit(1); // 자식 프로세스 종료

}

// 세 번째 자식 프로세스 생성

if (fork() == 0) { // 자식 프로세스인 경우

execl("/bin/echo", "echo", "this is", "message three", (char \*)0); // echo 명령어를 실행하여 "this is message three" 출력

fprintf(stderr, "exec error\n"); // execl 함수가 실패한 경우 에러 메시지 출력

exit(1); // 자식 프로세스 종료

}

printf("Parent program ending\n"); // 부모 프로세스에서 출력되는 메시지

exit(0); // 부모 프로세스 종료

}

2. 실행결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

<execve>

1. 소스코드

**[ssu\_execve.c]**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main(void) {

// 실행할 프로그램의 이름과 전달할 인자들을 지정한 배열

char \*argv[] = {

"ssu\_execl\_test\_1", "param1", "param2", (char \*)0

};

// 실행할 프로그램에 전달할 환경 변수들을 지정한 배열

char \*env[] = {

"NAME=value", // NAME 환경 변수 설정

"nextname=nextvalue", // nextname 환경 변수 설정

"HOME=/home/seyeon", // HOME 환경 변수 설정

(char \*)0 // 배열의 끝을 나타내는 NULL 포인터

};

printf("this is the original program\n"); // 프로그램 시작을 알리는 메시지 출력

execve("./ssu\_execl\_test\_1", argv, env); // ssu\_execl\_test\_1 프로그램을 실행하고 인자와 환경 변수를 전달

printf("%s\n", "This line should never get printed\n"); // execve가 성공적으로 실행되면 이 코드는 실행되지 않음

exit(0); // 프로그램 종료

}

2. 실행결과

텍스트, 스크린샷, 멀티미디어, 소프트웨어이(가) 표시된 사진

자동 생성된 설명