20185309 황명원

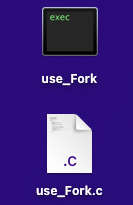
과제2 : fork 함수 사용

1. 기본 과제

1)바탕화면으로 cd 한뒤 .c 파일을 만들고 실행함수명을 만드는 과정

텍스트이(가) 표시된 사진

자동 생성된 설명

(파일생성)

2)use\_Fork.c의 소스코드

#include <sys/types.h>  
#include <stdio.h>  
#include <unistd.h>  
#include <sys/wait.h>  
  
int main()  
{  
 pid\_t pid;  
 /\* fork a child process \*/  
 pid=fork();  
 if(pid<0){ //error occurred  
 printf(stderr,"Fork Failed");  
 return 1;  
 }  
 else if(pid == 0){ // child process  
 execlp("/bin/ps","ps","-al",NULL);  
 }  
 else{ //parent process  
 //parent will wait for the child to complete  
 wait(NULL);  
 printf("Child Complete");  
 }  
 return 0;  
}

3) 실행결과

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 도전과제
2. fork 함수를 사용해서 프로세스를 4개 생성한다.

-> 우선 프로세스 4개를 생성하기위해 while문을 이용하여 작성했습니다.

\*소스코드

int main() {  
 int count = 0; //프로세스 실행수

pid\_t pid; 자식 프로세스의 pid 배열   
  
 while(count < 4) { //프로세스를 while문을 이용해서 진행  
  
 pid = fork();//fork 생성과 동시에 pid 값 저장  
  
 if(pid < 0) { //에러일 경우  
 return -1;  
 }  
 else if(pid== 0) {//자식 프로세스  
 exit(0);  
 }  
 count++; //다음 프로세스도 실행하기위해 count=count+1 해주기  
  
 }  
  
  
 return 0;  
}

1. 부모 프로세스는 자식 프로세스의 pid를 배열("int pid\_array[4]")에 보관한다.

->배열에 보관하기위해 배열을 생성하고 fork()를 실행과 동시에 pid값을 저장시켰습니다.

\*소스코드

int main() {  
 int count = 0; //프로세스 실행수  
 int pid\_array[4];//자식 프로세스의 pid 배열  
  
 while(count < 4) { //프로세스를 while문을 이용해서 진행  
  
 pid\_array[count] = fork();//fork 생성과 동시에 pid 값 저장  
  
 if(pid\_array[count] < 0) { //에러일 경우  
 return -1;  
 }  
 else if(pid\_array[count] == 0) {//자식 프로세스

exit(0);  
 }  
 count++; //다음 프로세스도 실행하기위해 count=count+1 해주기  
 }  
  
 return 0;  
}

1. 자식 프로세스는 생성후에  sleep()함수를 이용하여 5초에서 20초 사이에서 랜덤하게 sleep한다.

-> 5초에서 20초 사이로 랜덤값을 부르는 rand()를 사용했고 동시에 매번 실행할때마다

랜덤값도 달라지게 하기위해 srand(time(Null))도 사용했습니다.

\*소스코드

int main() {  
 int count = 0; //프로세스 실행수  
 int pid\_array[4];//자식 프로세스의 pid 배열  
 int randnum[4]; //랜덤 값들을 저장할 배열  
 srand(time(NULL));//랜덤 값을 매 실행마다 다르게 해주기 위한 코드  
  
 while(count < 4) { //프로세스를 while문을 이용해서 진행  
  
 randnum[count]=rand() % 16+5; //시작할때 랜덤 값 부여  
 pid\_array[count] = fork();//fork 생성과 동시에 pid 값 저장  
  
 if(pid\_array[count] < 0) { //에러일 경우  
 return -1;  
 }  
 else if(pid\_array[count] == 0) {//자식 프로세스  
 sleep(randnum[count]); //랜덤값으로 sleep  
 exit(0);  
 }  
 count++; //다음 프로세스도 실행하기위해 count=count+1 해주기  
 }  
 return 0;  
}

1. 잠에서 깨어난 후에 exec 함수를 사용해서 ps, ls, df, cal를 각각 수행시킨다.

-> 잠에서 깨어난 후에 exec함수를 사용하여 ps,ls,df,cal 을 수행시키기 위해

switch 문으로 나누어 count수마다 한가지씩 실행되게 만들었습니다.

\*소스코드

int main() {  
 int count = 0; //프로세스 실행수  
 int pid\_array[4];//자식 프로세스의 pid 배열  
 int randnum[4]; //랜덤 값들을 저장할 배열  
 srand(time(NULL));//랜덤 값을 매 실행마다 다르게 해주기 위한 코드  
  
 while(count < 4) { //프로세스를 while문을 이용해서 진행  
  
 randnum[count]=rand() % 16+5; //시작할때 랜덤 값 부여  
 pid\_array[count] = fork();//fork 생성과 동시에 pid 값 저장  
  
 if(pid\_array[count] < 0) { //에러일 경우  
 return -1;  
 }  
 else if(pid\_array[count] == 0) {//자식 프로세스  
  
 sleep(randnum[count]); //랜덤값으로 sleep  
  
 switch (count) { //각각의 exec를 실행  
 case 0:execlp("/bin/ps", "ps", NULL);  
 break;  
 case 1:execlp("/bin/ls", "ls", NULL);  
 break;  
 case 2:execlp("/bin/df", "df", NULL);  
 break;  
 default:execlp("/usr/bin/cal", "cal", NULL);  
 }  
 exit(0);  
 }  
 count++; //다음 프로세스도 실행하기위해 count=count+1 해주기  
 }  
 return 0;  
}

1. 자식 프로세스가 종료하면 부모 프로세스는 배열에 저장된 자식 프로세스의 아이디를. 출력시키고 종료한다.

->부모 프로세스에서는 자식프로세스가 모두 종료될때까지 기다리게 하기위해 while(count<4) 안에 while(wait(NULL)>0); 을 넣었고 while(count<4)를 다 수행하고 빠져 나올시 for문을 이용해 자식프로세스의 pid값을 출력 시키도록 하였습니다.(추가로 저는 랜덤값들도 배열로 저장하여 각각 몇초 동안 sleep했는지 알기위해 같이 출력하는 코드를 작성했습니다.)

\*완성된 소스코드

#include <stdio.h>  
#include <unistd.h>  
#include <stdlib.h>  
#include <sys/types.h>  
#include <sys/wait.h>  
#include <time.h>  
  
int main() {  
 int count = 0; //프로세스 실행수  
 int pid\_array[4];//자식 프로세스의 pid 배열  
 int randnum[4]; //랜덤 값들을 저장할 배열  
 srand(time(NULL));//랜덤 값을 매 실행마다 다르게 해주기 위한 코드  
  
 while(count < 4) { //프로세스를 while문을 이용해서 진행  
  
 randnum[count]=rand() % 16+5; //시작할때 랜덤 값 부여  
 pid\_array[count] = fork();//fork 생성과 동시에 pid 값 저장  
  
 if(pid\_array[count] < 0) { //에러일 경우  
 return -1;  
 }  
 else if(pid\_array[count] == 0) {//자식 프로세스  
  
 sleep(randnum[count]); //랜덤값으로 sleep  
  
 switch (count) { //각각의 exec를 실행  
 case 0:execlp("/bin/ps", "ps", NULL);  
 break;  
 case 1:execlp("/bin/ls", "ls", NULL);  
 break;  
 case 2:execlp("/bin/df", "df", NULL);  
 break;  
 default:execlp("/usr/bin/cal", "cal", NULL);  
 }  
 exit(0);  
 }  
 count++; //다음 프로세스도 실행하기위해 count=count+1 해주기  
 while(wait(NULL)>0); //모든 자식 프로세스끝날때까지 부모프로세스는 기다리기  
 }  
 for(int i=0 ; i<4 ; i++){  
 printf("자식 프로세스[%d] 종료! , child %d\n",i,pid\_array[i]);//자식프로세스 아이디 출력  
 printf("randnum[%d] : %d \n",i,randnum[i]); //각 자식 프로세스들 실행 됫을때의 얼마나 sleep 했는지 확인하기 위한 코드  
 }  
 return 0;  
}

-도전과제의 .c파일 작성(challenge.c) 하는 과정

텍스트이(가) 표시된 사진

자동 생성된 설명

-수행화면

텍스트이(가) 표시된 사진

자동 생성된 설명

(실행일자 기준 2022.03.24 밤 11시경)