**REPORT**



|  |  |
| --- | --- |
| **과목명** | 유닉스 프로그레밍 |
| **학과** | 컴퓨터공학과 |
| **학번** | 12161658 |
| **이름** | 조재민 |
| **제출일자** | 20.11.01 |

개요:

이번 3차 유닉스 프로그래밍 과제로는, 쉘을 구현하는 것으로 첫번째로 change directory와 background program이 정상적으로 작동하지 않는 것을 해결하는 과제 였습니다.

**요구사항:**

1. **‘cd’ 명령이 제대로 작동하지 않는 버그를 수정**
2. **백그라운드 실행을 구현**
3. **‘exit’ 구현**

**Change Directory**:

기본 제공되는 template code 에서 ls, mkdir, ps 등 리눅스의 기본 명령어 들은 정상 작동 하지만 cd가 작동하지 않았습니다. 이유는 fork() 함수를 통해 자식 프로세스로 작동할때 change directory를 하면 부모 cwd (current working directory)와 자식의 cwd에 차이가 생기는 문제였습니다.

**텍스트이(가) 표시된 사진

자동 생성된 설명해결방안:** ‘cd’ 명령어를 main 함수 내에 fork() 함수 활용 전 input을 strcmp로 확인하여 change directory를 처리한다.

While문으로 shell programming이 끝나지 않고 계속 실행 되며, 매번 input을 받으면 change directory, 즉 cd 명령어가 입력됐는지를 확인합니다. 이는 fork() 함수와 if문으로 구별해놓아 부모, 자식간의 cwd가 달라지는 문제를 해결하였습니다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**구현된 change directory 함수 입니다.

Argc 가 1, 즉 인자 없는 cd 가 들어오면 home directory로 가도록 getenv 함수를 활용하였습니다.

Cd ~ 활용 역시 정상 작동 하도록 두번째 인자에 예외처리를 하였습니다.

두번째 인자로 들어온 directory가 정상적인 directory라면 해당 directory로 이동합니다.

**결과:**

텍스트이(가) 표시된 사진

자동 생성된 설명

기존 에러인 main: No Such File or Directory 가 아닌

Cd 명령어가 정상적으로 작동하는 것을 확인 하였습니다.

**Background**:

Command line에서 뒤에 & 이 붙으면 background에서 수행하여야 하므로, 고아 프로세스의 생성 원리를 이용하였습니다.

& 입력이 들어왔을때 해당 실행을 background 에서 한다는 것을 구분하기 위하여 background 변수를 생성하여 & 입력 시 해당 값을 바꿔 주었습니다. 또한 & 가 입력으로 들어오면 문자를 ‘\0’ 로 변경해 주었습니다.

그 다음 main 함수의 switch case 에서 background 값이 0, 즉 부모 프로세스라면 waitpid 함수를 호출하여 자식 프로세스가 종료 되었을때 거두어 주는 함수를 활용하였습니다. 이렇게 하여서 자식 프로세스가 좀비 프로세스가 되는것을 방지하였습니다.

**해결방안:**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

If문으로 입력에 & 들어왔는지 확인 및 문자열 변경

부모 프로세스라면 waitpid 함수를 호출

**결과:**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

Sleep 10 사용시 10초 대기 후 shell에 재입력 프로세스가 실행 하지 않고 background 에서 돌아가는 것을 확인 하였습니다.

**좀비 프로세스 생성 이유:**

좀비 프로세스란 fork, exec 를 활용해 자식 프로세스가 별도의 작업을 수행하고 부모 프로세스는 자식을 기다리지 않을 때의 자식 프로세스를 의미합니다. & 가 입력으로 들어왔을 경우 자식 프로세스가 background에서 수행 되는데 해당 프로세스를 별도의 프로세스로 구분 짓고 부모 프로세스를 계속해서 수행 하였기 때문에 해당 자식 프로세스의 수행 종료시 부모 프로세스가 waitpit 함수로 거두어 주어야 합니다. 그렇지 않으면 생성된 자식 프로세스가 종료 됐을경우 할당된 영역을 거두어준 부모가 없기 때문에 소위 ‘찌꺼기’ 처럼 남아 있을 것입니다.

**Exit 구현:**

기본 제공되는 template 코드에서 myshell 프로그램이 종료 될려면 강제 종료 입력인 ^c 입력만이 작동하였습니다. While 문으로 계속해서 실행되는 프로세스 이기 때문에 별도 종료가 없으면 계속해서 실행 되었습니다. 이 부분을 해결 하기위해 입력으로 exit 이 들어오면 프로세스를 종료하도록 설계 하였습니다.

해결방안:

Exit 입력이 들어왔는지 strcmp 함수를 활용하여 비요해 줍니다. 해당 값이 0 이라면, 즉 exit을 입력하였다면 프로세스를 종료 exit(1)을 해주었습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

앞서 cd 입력과의 비교가 필요하기 때문에 else if 문으로 시작하였습니다. 또한 밑의 else 문은 현재 cd, exit 의 입력이 들어오지 않아 fork 를 호출하는 함수로 진입합니다.

**결과:**

텍스트이(가) 표시된 사진

자동 생성된 설명

Exit 입력시 정상적으로 메시지 출력 (myshell exit…) 과 함께 종료되었습니다.