Shell program_Assignment3

2019062833 김유진

1. eval

```
char *cmdArr[MAXARGS];
int bg, b_ch, e_ch;
pid_t pid;


bg = parseline(cmdline, cmdArr);

if (cmdArr[0] == NULL)  return;

b_ch = builtin_cmd(cmdArr);
```

-명령을 받아서 배열안에 저장.
-명령이 없는 경우 바로 return.
-내장된 명령이면 즉시 실행.

```
    if (!b_ch) {
        if ((pid = fork()) < 0) {
            exit(1);
        }

        else if (pid > 0) {
            if (!bg) {
                addjob(jobs, pid, FG, cmdline);
                waitfg(pid);
            }
            else {
                addjob(jobs, pid, BG, cmdline);
                printf("[%d] (%d) %s", pid2jid(pid), pid, cmdline);
            }
        }
```

-내장된 명령이 아니었다면 아래 코드를 실행.
-fork 함수로 현재 실행중인 프로세스를 복사해서 다른 프로세스를 실행.
-fork 실패 시 종료.
-부모 프로세스인 경우 addjob을 하고 fg일 경우 wait

```
        else if (pid == 0) {
            if (setpgid(0, 0) < 0) {
                exit(1);
            }

            e_ch = execve(cmdArr[0], cmdArr, environ);

            if (e_ch < 0) {
                printf("%s: Command not found\n", cmdArr[0]);
                exit(1);
            }
        }

    }
    return;
```

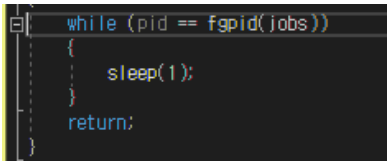
-자식 프로세스인 경우 자식 pid와 동일한 그룹 id에 넣음.
-실행 가능한 파일인 filename의 실행 코드를 현재 프로세스에 적재하여 기존 실행코드와 교체하여 새로운 기능으로 실행.
-실패 시 에러메시지 출력 후 종료.

2. builtin_cmd

```
 */
int builtin_cmd(char **argv) //TODO
{

    if (!strcmp(argv[0], "jobs")) {
        listjobs(jobs);
        return 1;
    }

    else if (!strcmp(argv[0], "quit")) {
        exit(0);
    }

    else if (!strcmp("&", argv[0])) {
        return 1;
    }

    else if (!strcmp(argv[0], "bg") || !(strcmp(argv[0], "fg"))) {
        do_bgfg(argv);
        return 1;
    }

    return 0;
}
```

-내장된 명령어인지 판별하고 맞으면 해당 명령 수행.
-명령어 “jobs”, ”quit”, ”bg”, ”fg”에 대한 처리.
-&가 명령어에 있다며 background 실행.
-내장된 명령어가 아니었을 경우 0 리턴

3. waitfg



```
while (pid == fgpid(jobs))
{
    sleep(1);
}
return;
}
```

-foreground 작업하는 동안에는 wait