이번 과제는 운영체제의 'Shell'을 구현하는 과제였습니다. 크게 총 5가지 기능을 구현해야 했습니다.

먼저 첫 번째는 'Prompt' 명령어를 입력하면, 그 뒤에 입력한 문자/숫자로 Prompt의 모양을 변형시키는 것이었습니다. Char __prompt 배열에 strcpy() 함수를 사용하여 입력 받은 문자열을 복사하여 저장하여 __prompt의 값을 변화시켰습니다.

두 번째는 cd 명령어를 구현하는 것이었습니다. Cd 뒤에 입력한 디렉토리로 이동을 하는 기능을 구현해야 했습니다. 이는 chidir() 함수에, run_command()의 매개변수로 입력되는 tokens[1]의 문자열을 매개변수로 입력해 올바른 기능을 수행하도록 했습니다. 이때 'cd ~'는 home directory로 이동해야 하므로, 따로 if()문으로 분리하여 수행하도록 했습니다.

세 번째는 반복 기능을 구현하는 것이었습니다. 'for num command' 순서로 입력을 받으면, 명령어의 기능을 숫자만큼 반복하도록 하는 것이었습니다. For command를 입력한다면 Tokens[1]자리에 숫자가 오므로, string을 int로 형 변환을 해주는 atoi()함수를 통해 숫자 값을 얻어냈고, 그 숫자의 값만큼 for 문을 실행시켜 주었습니다. 이때 for num for num command 형식으로 입력을 받는 경우가 존재하기 때문에, tokens[1], tokens[3] 이 for command이고, tokens[2], tokens[4]이 number이므로, 2번째, 4번째 token을 strcpy()함수를 이용하여 1번째, 3번째 자리로 옮겨준 후, run_commnad() 함수를 다시 호출하는 방법을 사용했습니다.

위의 cd, for, prompt 기능은 모두 built-in command로, strcmp()함수를 이용하여, 사용자의 입력과 명령 command를 직접적으로 비교하여 알맞은 if()문으로 들어가 맞는 기능을 수행하도록 했습니다.

네 번째 기능은 time-out 기능입니다. 같은 built-in command로서, strcmp() 함수를 사용하여 첫 입력이 timeout인지를 비교한 후 if()문에 들어가 기능을 수행하도록 했습니다 Timeout기능은 external program이 'timed-out'이 됐는지 안 됐는지를 알 수 있게 해주는 역할을 합니다. 먼저 set_timeout() 함수에서 매개변수로

들어온 timeout변수로 __timeout변수를 초기화시킵니다. 즉 현재 time limit값을 초기화시킵니다. 이 함수를 이용하였고, 만약 timeout command 하나만 들어온다면 현재 time limit 값을 출력해 주었고, 뒤에 num이 함께 입력된다면 set_timeout() 함수를 이용하여 현재의 time limit값을 입력 받은 값으로 초기화 시켜주었습니다. 여기까지는 shell 안에서 작동하는 built-in 기능들이었고, 저는 External program들이 수행될 수 있게 해야 했습니다.

External program들은 shell 내에서 동작하는 것이 아니기 때문에 fork를 통해 똑같은 process 공간을 생성했습니다. 그러나 하는 역할이 달라야 하므로, execvp() 함수를 이용하여 tokens[1]에 있는 command를 수행할 수 있도록 했습니다. 이때 execvp() 함수의 return 값이 0보다 작다면 존재하지 않는 command를 입력받은 것이므로 No such file or directory 라는 출력문이 나오도록 했습니다. 이때 parent process는 child process의 status를 wait하여, 만약 time-limit 시간보다 작업 시간이 오래 걸린다면, kill을 통해 process를 종료하는 기능을 수행해야 합니다. 이를 위해 먼저 timeout-handler를 만들고, alarm()함수에 timeout시간 값을 매개변수로 입력하여, timelimit 시간이 지나면, signal이 발생하도록 했고, handler를 이용하여 현재 입력 받은 command가 timeout되었다는 메시지를 출력하도록 했습니다. 메시지를 출력한 후, kill(pid, SIGKILL) 함수를 이용해 process를 종료했습니다. 이때 pid값은 pid = fork() 로 얻은 값입니다. Parent Command는 child process의 pid값을 return 받기 때문에 kill() 함수에 매개 변수로 입력합니다. SIGALRM signal을 sigaction(SIGALRM, &sa, NULL)을 통해 timeouthandler을 실행시킵니다. Timer signal을 alarm()으로부터 입력받습니다.

이번 과제를 통해 이론으로만 배웠던 shell이 어떻게 동작하는지 간접적으로 알 수 있었습니다. 하지만 signal부분과 handler부분에서 완벽한 구현을 하는 중 큰 어려움을 겪었습니다. 그러나 저의 실력이 상승하고, 지식이 쌓이는 과제였다고 생각합니다. 무엇보다 fork()에 대한 이해도를 높일 수 있었습니다.