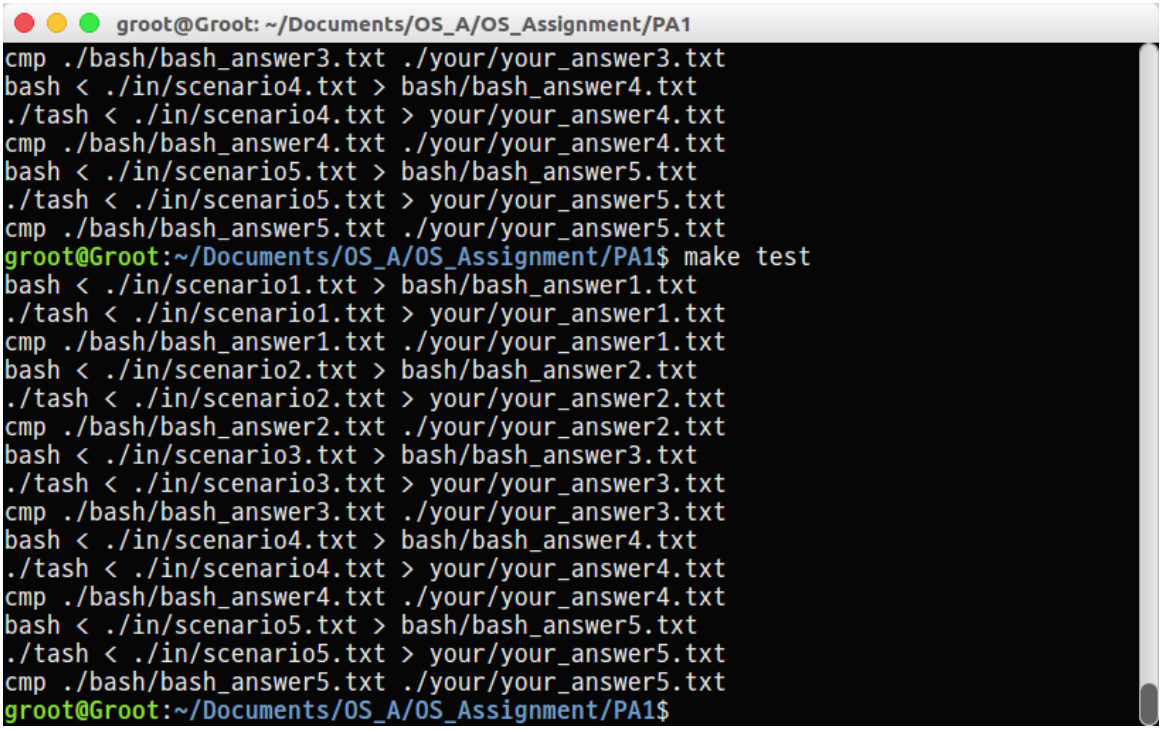
과제의 수행 정도

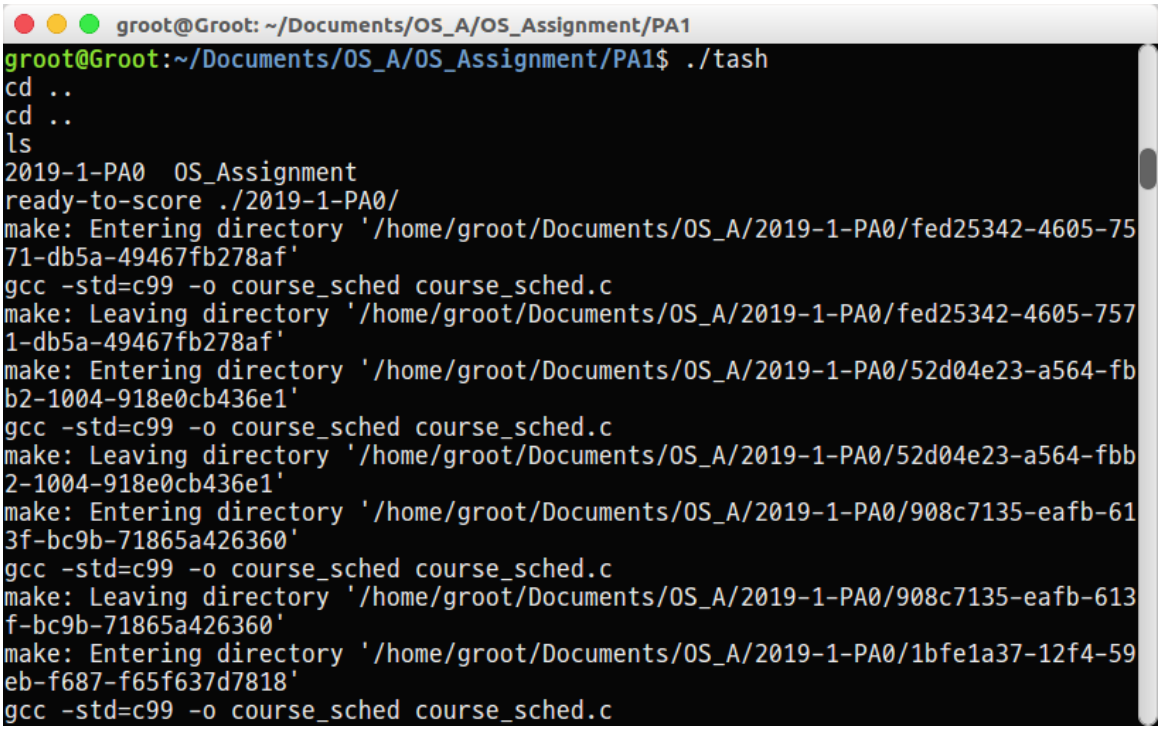
시나리오 1부터 5까지의 입력 값을 bash에서 수행한 결과와 구현한 프로그램의 결과와 일치하는 것을 확인하였습니다. 이전에 랜덤으로 test가 성공했던 점을 토대로 make test를 여러 번 수행한 결과 확실하게 성공함을 확인할 수 있었습니다.

built-in command인 "pwd", "cd"를 구현하였고, 나머지 명령어 ls, make, echo. grep은 exec계열 함수를 사용하여 구현하였습니다. File redirection은 dup2( )함수를 이용하여 구현하였습니다.

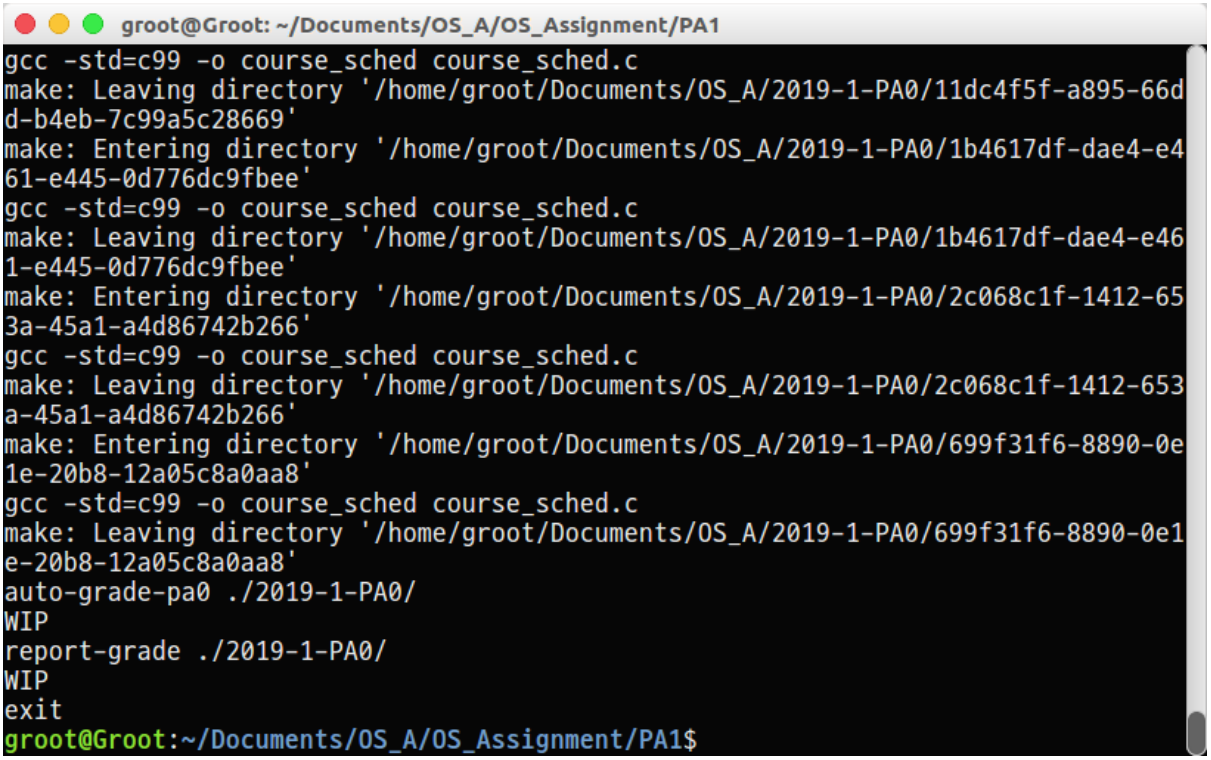
시나리오 자가 채점



채점 보조 기능 수행 결과



2019-1-PA0 디렉터리(가짜 과제 파일들이 있는 곳)로 경로를 옮기고 alias되어 있는 python script를 실행시키는 명령어를 입력하여 채점 보조 기능을 수행하도록 하였습니다.



과제를 통해 배운 것

- unistd와 fcntl 헤더파일에 있는 함수들을 알게 되었습니다.
- 각 명령어를 구현하는데 필요한 함수들이 존재하는 것도 알게 되었습니다.
  현재 디렉터리 경로를 받아오는 getcwd( ), 환경변수를 값으로 받아오는 getenv( ), 현재 경로를 변경해주는 chdir( ), 실행파일을 실행시켜주는 exec 계열 함수들, file descriptor를 복사해주는 dup2( )함수를 알게 되었습니다.
- dup2( )를 이용해 "문자열 > 파일"에서 파일을 출력 역할(file descriptor of stdout)로 변경해주어 printf( ) 만으로 파일에 문자열을 쓸 수 있다는 것을 알았습니다. 이는 또한 입력 역할도 가능하게 한다는 것을 알았습니다.
- exec계열 함수들을 공부하면서 execl(리스트 형식: 나열), execv(벡터 형식: 배열)에
  p(path)와 e(environment)를 함께 사용해(execlp, execle와 같이) 쓰임새에 맞게 사용할 수 있는 것을 알았습니다. 심지어 각 실행파일의 옵션들도 넣어주어 실행시킬 수 있다는 것과 마지막에 NULL을 넣어 옵션이 끝났음을 확인하는 것까지 수행한다는 것에 놀랐습니다.
- exec 계열 함수들이 어떻게 실행되는지 몇 번의 시도를 통해서 프로세스의 생성, 변경 그리고 수행이 끝날 때까지의 과정을 이해할 수 있었습니다. 이는 특히 명령어 구현 후, make test를 통해 여러 번 test할 경우 처음엔 무작위로 성공되는 문제가 생겨 입력 버퍼를(fgets함수는 new line인 개행 문자까지 받는 것 또한 알게 되었습니다.) 프로세스 생성 이전에 비워주고 waitpid가 아닌 wait(NULL)을 사용해 자식 프로세스(exec계열로 명령어를 수행 중인)가 끝날 때까지 기다리도록 하여 파일에 순서대로 쓰여지도록 하여 문제를 해결함으로써 익혔습니다. 파일에 순서대로 쓰이지 않는 것이 부모 프로세스가 자식 프로세스의 수행을 기다리지 않고 다른 자식 프로세스를 생성하여 수행하는 과정에서 뒤섞인 것 같다고 생각했기 때문입니다.

과제에 대한 피드백

- 명령어가 추가될수록 코드의 길이가 길어져 이를 패턴 별로 나누어 간단하게 할 필요성이 있습니다. 예를 들어, exec계열 함수로 구현된 명령어는 패턴이 비슷합니다.
- 단순히 시나리오에 맞춰 명령어를 구현한 것이 아닌 좀 더 범용적인 즉 실제 bash에서 명령어를 수행했을 때와 같이 구현하도록 하였습니다. 특히 cd 명령어의 경우 "cd ~"까지 수행할 수 있도록 하였고, echo 역시 "echo '문자열'"과 "echo 문자열" 등 최대한 실제 명령어에 가깝게 구현하려고 노력했습니다.
  하지만, 명령어에 존재하는 수많은 옵션들을 구현하지 못했다는 점과 alias의 명령어는 실제로 bash_aliases에 명령어에 대한 별칭을 등록하여 수행하게끔 시도했지만, 등록 후에 동기화를 어떻게 해야 할지 구현하지 못해 alias 명령어가 아닌 alias 같은 명령어만 구현하여 아쉬움이 많이 남습니다.
- File redirection 부분도 각 명령어 마다 수행될 수 있게 구현되도록 하는 것이 좋습니다.
- 입력 받은 문자열을 명령어 / 옵션 등 원하는 부분으로 파싱해주는 기능을 따로 구현하는 것이 좋습니다. 현재 프로그램은 단순히 각 명령어에 따라 파싱하는 작업을 수행하기 때문에 코드가 길어지고 비효율적입니다.