1. Outline how programs are launched and how arguments are passed(external program 구현방법)

"for", "prompt", "timeout", "cd", "exit"는 built-in command로 각각 명령어에 해당하게 구현하였다. 반면 external program을 실행할 때는 fork()와 exec()를 사용하여 진행하였다. 그 안에서 3가지로 나누어 구현하였다.

execv(const char *path, char *const argv[])는 첫번째 인자에 경로를 두번째 인자에 문자열 배열 매개변수를 통해 프로그램 인수를 받는다. 이를 사용하여 external program을 구현하였다.

external program

- 현재 경로까지 해서 실행

"./toy"일 경우 toy라는 실행파일을 실행해야 하므로 getcwd(PATH, 2048) 현재 경로 뒤에 추가하는 코드를 구현한 후 execv()로 PATH와 tokens를 모두 넘긴다.

- bin 붙이면 경로 따로 안 붙여도 바로 실행

execv()로 PATH와 tokens를 모두 넘긴다.

- bin 없을 때 bin 경로 붙여서 실행

"/bin/"의 경로를 붙인 후 execv()로 command와 tokens를 모두 넘긴다.

2. How the timed-out feature is implemented(timed-out 구현방법)

- alarm(__timeout)

부모에서 알람 호출을 설정한다. 시간이 되면 자기 프로세스에 SIGALRM을 보낸다. SIGALRM은 프로세스를 종료한다. 하지만 우리는 signal_handler로 SIGALRM으로 종료하지 않고 처리한다.

- sigaction(SIGALRM, &act, NULL)

Sigaction이라는 system call을 날리면 SIGALRM을 act에 맞춰서 시그널을 처리한다. 옛날 정보는 없으므로 NULL을 넣는다.

- signal_handler()

name을 받아서 timeout 되었다는 문구를 출력한다. 그리고 kill이라는 함수(kill(child_pid), SIGKILL)로 child_pid라는 프로세스를 가진 애한테 SIGKILL을 보낸다.

3. How the “for” built-in command is implement(for 구현방법)

명령어를 치면 한 번은 실행되어야 하기 때문에 run_command가 실행되면 큰 for문 안에 넣고 num만큼 실행하게 한다. (초기 num의 값은 1이다.)

- “for” 명령어 분리

Shell에서 “for”이라는 command를 치면 뒤에 나오는 숫자만큼 반복 실행되어야 한다. “for” 뒤에는 숫자가 나온다. 따라서 처음 tokens[0]이 “for”라면 “for” 다음은 숫자가 무조건 나오므로 이를 num이라는 변수에 저장한다.

- “for” 명령어 분리가 끝나면

나머지 명령어를 tokens[j]에 차례대로 복사한다. 그리고 num만큼 shell에서 command에 맞게 external program이나 exec를 사용하여 실행한다.

4. Lessons learned if you have any

- exec() 함수의 사용법

execl() : 시스템 호출을 할 때 명령줄 인수를 하나씩 나열하고 NULL은 인수 끝을 나타낸다.

execv() : 명령줄 인수를 하나씩 나열하지 않고 명령줄 인수 리스트를 포인터 배열로 만들어 이 배열의 이름을 전달한다.

- 경로 변경

chdir() : 프로세스는 현재 작업 디렉토리를 가지고 있는데 이 함수를 호출하면 현재 작업 디렉토리를 변경한다.

- 제약사항을 확인하지 못하고 처음에 일일히 명령어를 구분한 뒤 구현하였다. QnA게시판을 확인하다가 깨닫게 되어 코드를 전면 수정하게 되었다. 다음 과제를 진행할 때는 README.md 파일을 꼭 숙지하고 진행해야겠다.