과제 #2 : SSUShell 및 기본 명령어 구현

○ 과제 목표

- Linux에서 기본적으로 제공하는 bash 쉘과 유사한 간단한 SSUShell 구현
- 운영체제의 프로세스 관리 기본 개념 이해
- 시스템 호출 함수를 이용하여 간단한 프로그램 작성 기법 이해
  - ✓ Linux의 간단한 내장 명령어(echo, cat, sleep, ls, ps, top, grep 등) 기능 이해
  - ✓ Linux의 간단한 내장 명령어와 유사한 명령어(pps, ttop) 구현

○ 기본 지식

- Linux의 프로세스 관련 시스템 호출 함수 (fork, exec, exit, wait 등)

○ 과제 내용

- 0. Linux 설치
  - ✓ 배포판 : Ubuntu (18.04 이상) 또는 Fedora (5.3 이상) 권장
  - ✓ 상세한 설치 방법은 인터넷 자료 활용
    - Dual Booting (Windows 및 Linux 동시 설치)
      - ☞ ※ 참고 : https://coding-factory.tistory.com/494
    - 가상 머신 (Virtual Box pro 또는 Vmware 등을 이용)으로 Linux 설치
      - ☞ ※ 참고 :https://hiseon.me/linux/ubuntu/ubuntu-virtualbox-install/

- 1. Linux 내장 명령어를 실행하는 간단한 쉘 (SSUShell) 구현
  - ✓ ssu_shell.c skeleton 코드를 완성
    - 입력 (대화식 또는 배치식 모드로)을 읽고 입력을 토큰화
      - ☞ ./ssu_shell : 대화식 모드
      - ☞ ./ssu_shell commands.txt : 배치식 모드
  - ✓ SSUShell은 사용자 입력 -> fork(2)를 사용하여 하나 이상의 자식 프로세스를 생성 -> 자식 프로세스들로부터 exec()를 호출하여 사용자 명령을 실행 -> wait(2)를 사용하여 죽은 자식을 거둠
  - ✓ SSUShell은 ls, cat, echo 및 sleep과 같은 모든 간단한 Linux 내장 명령어 실행 가능해야 함
    - Linux의 쉘 명령어을 구현하는 system(3) 같은 라이브러리 기능을 사용해서는 하면 안 됨
    - SSUShell의 모든 명령어는 리디렉션 또는 기타 특수한 경우가 없는 Linux 제공 명령어와 직접 구현 명령어 pps, ttop
      - ☞ Linux가 제공하는 기본 명령어는 실행되어야 하나, pps와 ttop은 별도로 구현하고 구현한 명령어가 실행되어야 함.
  - ✓ SSUShell의 프롬프트 : $
  - ✓ SSUShell은 “대화식” 또는 “배치(일괄처리)식“ 중 하나로 실행되어야 함
    - SSUShell은 명령어의 인자가 주어지 않으면 사용자의 입력을 대화식으로 받아들이고 실행
    - SSUShell은 명령의 배치 파일이 프로그램에 명령 줄 입력으로 제공되면 배치 파일의 모든 명령을 차례로 실행. 예제로 샘플 배치 파일 commands.txt 제공. 예제로 제공된 commands.txt 뿐만 아니라 모든 내장 Linux 명령을 처리 가능해야 함
    - SSUShell은 이전 명령 실행이 완료된 후에만 사용자 입력을 위해 리턴(또는 일괄처리식의 다음

명령으로 이동)
- 대화식 모드에서 사용자가 셸을 종료하기 위해 Ctrl + C를 누를 때까지 무기한 실행을 계속
- 배치식 모드에서 쉘은 배치 파일 끝에 도달하면 종료

✓ 실행할 명령어와 해당 인자의 입력에서 하나 이상의 공백으로 구분되어 있다고 가정하고 공백을 구분 기호로 사용 -> 입력 스트림을 "토큰화"
- "토큰화 된 명령을 실행하기 위해 ssu_shell.c 에 코드를 추가
- 입력 명령어는 1024자 이하, 토큰은 64개 이하 (각 토큰이 64자를 넘지 않아야 함)
- Linux 내장 명령어는 SSUShell의 명령어 인자로 호출되고 백그라운드 실행, I/O 리디렉션 같은 특수 실행 모드 없이 호출 가능해야 함. 단, 파이프(unnamed pipe) 기능은 수행되어야 함.
  ex) $ls ¦ more, pps ¦ grep bash, ls ¦ grep ¦ wc 등
- 입력 스트림에서 다른 특수 문자를 구문 분석 불필요 (파이프 제외)

☞ 사용자가 Linux 명령어에 잘못된 옵션을 제공했는지 확인 코드는 구현하지 않아도 됨
- 명령어에 대한 인자를 확인하거나 명령어가 있는지 여부 확인 코드는 구현하지 않아도 됨
- 단순히 사용자가 입력으로 제공하는 모든 명령어에 대해 exec를 간단히 호출하면 됨. 잘못된 명령어 또는 인자로 인해 Linux 명령어 실행이 실패하면 화면에 오류 메시지 프린트하고 다음 명령어로 이동

✓ SSUShell은 오류를 정상적으로 처리해야 함
- 빈(empty) 명령어(단순히 엔터를 누른 입력)은 단순히 쉘이 오류 메시지 없이 $ 프롬프트를 다시 표시
- 모든 잘못된 명령이나 기타 잘못된 입력에 대해 SSUShell은 "SSUShell : Incorrect command" 오류 메시지 출력 후, 사용자가 다음 명령어를 입력할 수 있게 해야 함

✓ 모든 명령어에 대해 SSUShell 이 생성한 자식 프로세스를 종료하고 회수해야 함

✓ 테스트 프로그램에서 ps 명령(pps가 아닌 기본 내장 명령어)을 사용하여 이 속성을 확인

✓ SSUShell에서 몇 가지 일반적인 Linux 제공 명령어를 실행하고 출력이 일반 Linux 쉘에서 나오는 결과와 일치하는지 확인하고 자식 프로세스가 종료 후 회수되어 시스템에 남아 있는 좀비 프로세스가 있는지 확인

✓ 분기 된 자식이 exec(2)를 호출하여 명령어를 실행하면 실행 파일이 완료된 후 자식 프로세스는 자동으로 종료됨. 그러나, 어떤 이유로 exec(2) 호출이 성공하지 못한 경우 SSUShell은 자식 프로세스가 적절하게 종료되었는지 확인해야 함

✓ 아무런 명령어도 실행하지 않았을 경우, 시스템에서 실행중인 SSUShell 프로세스가 하나만 있어야하며 다른 자식 프로세스는 없어야 함

- 2. 쉘 명령어 ttop, pps 구현

✓ SSUShell의 모든 명령어는 리디렉션 또는 기타 특수한 경우가 없는 Linux 제공 명령어와 직접 구현 명령어 pps, ttop로 구성
- Linux가 제공하는 기본 명령어는 실행되어야 하나, pps, ttop은 별도로 구현하고 구현한 명령어가 실행되어야 함

✓ ttop와 pps는 사용자 홈 디렉토리(pwd) 내 존재해야 함

✓ (1) ttop
- ttop 명령어 입력 시 기본 출력은 기본 내장 명령어 top와 유사한 기능을 갖고 있음

☞ interval 간격 (3초) 마다 화면을 갱신하여 정보가 출력됨

☞ top 명령어와 동일하게 키보드의 상하 화살표 키를 사용하여 목록에 대한 스크롤 제어가 가능

해야 함

☞ 상하 화살표키 입력 시 interval 시간에 상관없이 화면 정보가 갱신됨

☞ 명령어 결과 출력 시, 현재 터미널 크기에 맞게 글자 수가 한 행을 넘어가지 않도록 구현 (libcurses5-dev 패키지 사용 필요)

☞ 문자 'q' 입력 시 해당 명령어가 종료되며 쉘로 복귀하여 $ 프롬프트를 다시 표시

✓ (2) pps

- pps 명령어 입력 시 기본 출력은 기본 내장 명령어 ps와 유사한 기능을 갖고 있음

☞ 명령어 결과 출력 시, 현재 터미널 크기에 맞게 글자 수가 한 행을 넘어가지 않도록 구현 (libcurses5-dev 패키지 사용 필요)

- 옵션 a, u, x 구현

☞ 각각의 옵션에 대한 출력은 내장 명령어 ps의 a, u, x 옵션과 동일하며 기능은 man 페이지로 확인 (-a, -u, -x 옵션이 아님을 주의)

- 위 세 가지 옵션은 함께 사용 가능해야 함 (예. pps ux, pps aux 등)

○ 과제 수행 후 출력 결과 예

```
$ cat a.txt
this is oslab
2020 os project
$ cat a.txt | grep os | grep 2020
2020 os project
```

[그림 1] 다중 파이프 실행 예시

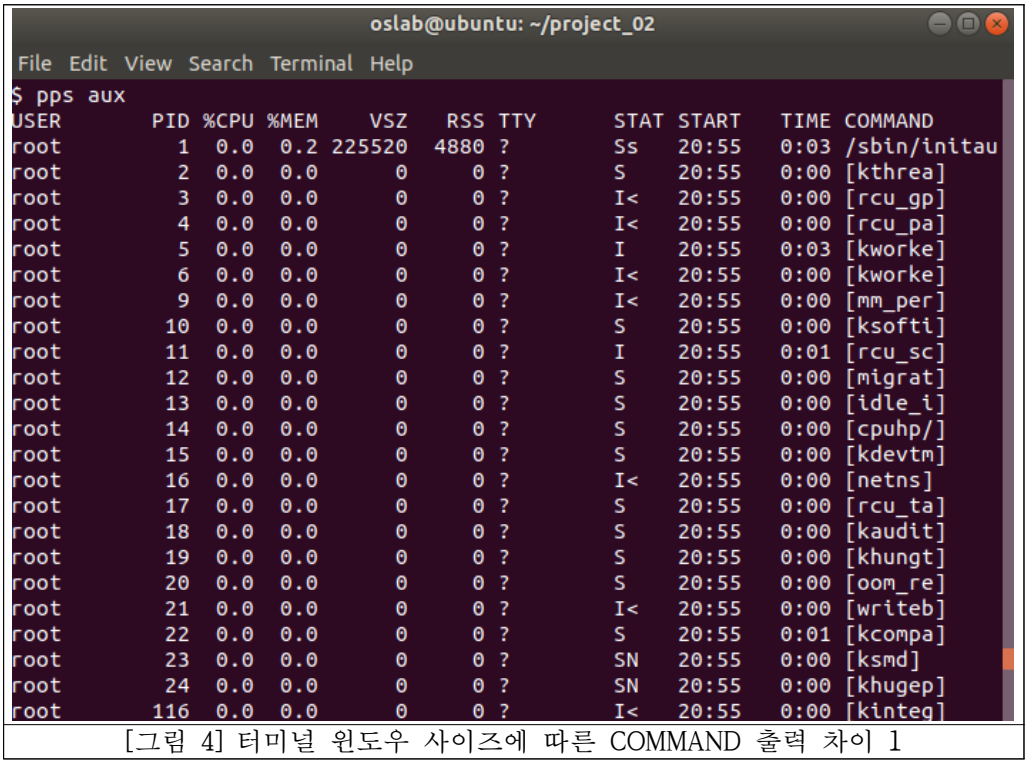
```
$ pps
  PID TTY          TIME CMD
 3395 pts/0    00:00:00 bash
 3406 pts/0    00:00:00 ./SSU_shell
 3407 pts/0    00:00:00 pps
```

[그림 2] pps 명령어 출력 예시

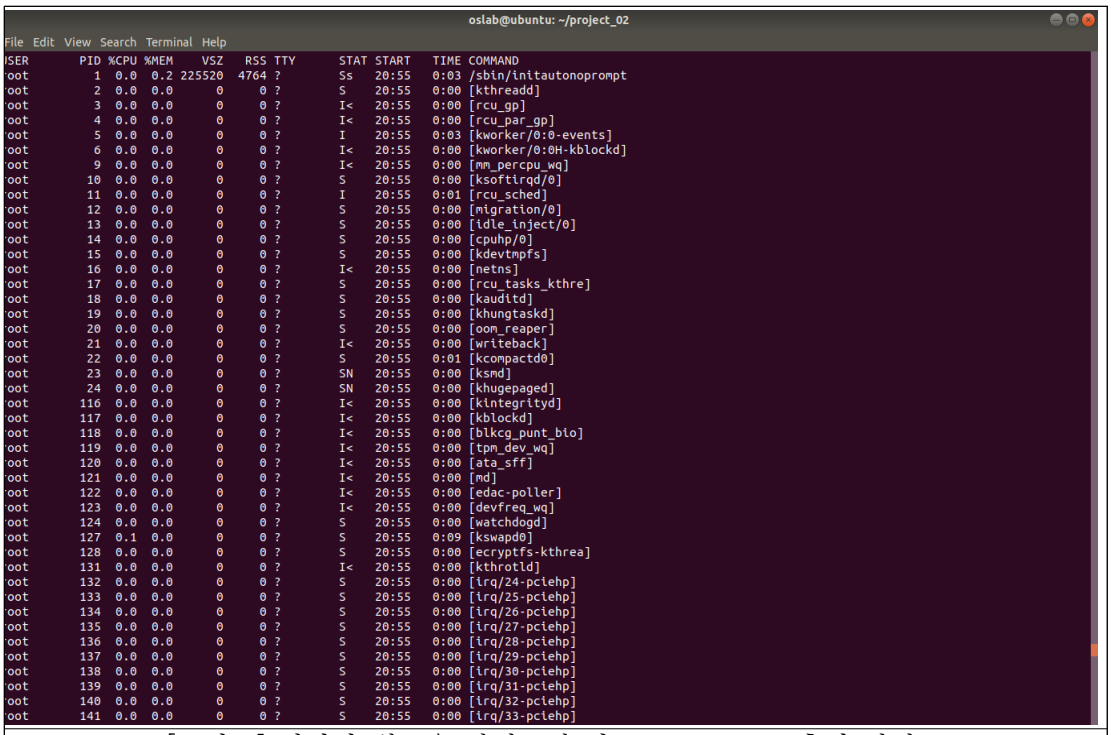
```
top - 22:41:09 up 1:45,  2 user,  load average: 0.84, 0.41, 0.30
Tasks:  386 total,    2 running,  316 sleeping,    0 stopped,    0 zombie
%Cpu(s)  99.3 us,   0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.7 si,  0.0 st
KiB Mem : 2006904 total,    71564 free,  1798304 used,  137036 buff/cache
KiB Swap:  969960 total,        0 free,   969960 used,    37812 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
    1 root      20   0  225520   2004     64 S   0.0  0.0  0:04.19 /sbin/initautonoprompt
    2 root      20   0       0      0      0 S   0.0  0.0  0:00.01 kthreadd
    3 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 rcu_par_gp
    5 root      20   0       0      0      0 I   0.0  0.0  0:03.71 kworker/0:0-eve
    6 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 kworker/0:0H-kb
    9 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 mm_percpu_wq
   10 root      20   0       0      0      0 S   0.0  0.0  0:00.62 ksoftirqd/0
   11 root      20   0       0      0      0 I   0.0  0.0  0:01.75 rcu_sched
   12 root      rt   0       0      0      0 S   0.0  0.0  0:00.05 migration/0
   13 root     -51   0       0      0      0 S   0.0  0.0  0:00.00 idle_inject/0
   14 root      20   0       0      0      0 S   0.0  0.0  0:00.00 cpuhp/0
   15 root      20   0       0      0      0 S   0.0  0.0  0:00.00 kdevtmpfs
   16 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 netns
   17 root      20   0       0      0      0 S   0.0  0.0  0:00.00 rcu_tasks_kthre
   18 root      20   0       0      0      0 S   0.0  0.0  0:00.00 kauditd
   19 root      20   0       0      0      0 S   0.0  0.0  0:00.02 khungtaskd
   20 root      20   0       0      0      0 S   0.0  0.0  0:00.00 oom_reaper
   21 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 writeback
   22 root      20   0       0      0      0 S   0.0  0.0  0:01.62 kcompactd0
   23 root      25   5       0      0      0 S   0.0  0.0  0:00.00 ksmd
   24 root      39  19       0      0      0 S   0.0  0.0  0:00.00 khugepaged
  116 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 kintegrityd
  117 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 kblockd
  118 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 blkcg_punt_bio
  119 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 tpm_dev_wq
  120 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 ata_sff
  121 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 md
  122 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 edac-poller
  123 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 devfreq_wq
  124 root      rt   0       0      0      0 S   0.0  0.0  0:00.00 watchdogd
  127 root      20   0       0      0      0 S   0.2  0.0  0:14.80 kswapd0
  128 root      20   0       0      0      0 S   0.0  0.0  0:00.00 ecryptfs-kthrea
  131 root       0 -20       0      0      0 I   0.0  0.0  0:00.00 kthrotld
  132 root     -51   0       0      0      0 S   0.0  0.0  0:00.00 irq/24-pciehp
  133 root     -51   0       0      0      0 S   0.0  0.0  0:00.00 irq/25-pciehp
  134 root     -51   0       0      0      0 S   0.0  0.0  0:00.00 irq/26-pciehp
```

[그림 3] ttop 명령어 출력 예시



[그림 4] 터미널 윈도우 사이즈에 따른 COMMAND 출력 차이 1



[그림 5] 터미널 윈도우 사이즈에 따른 COMMAND 출력 차이 2

○ 과제 제출 마감

- 2020년 9월 23일 (수) 23시 59분 59초까지 구글클래스룸으로 제출
- 1일 지연 제출마다 30% 감점. 4일 지연 제출 시 0점 처리 (이하 모든 설계 과제 동일하게 적용)

○ 필수 구현 (설명)

- 1. Linux 내장 명령어를 실행하는 간단한 쉘 (SSUShell) 구현

○ 배점 기준

- 1 : 30점
- 2 (1) : 30점
- 2 (2) : 40점