설계과제 2 개요 : SSU-Cleanupd

○ 개요

- 리눅스 시스템 상에서 사용자가 원하는 디렉토리를 자동으로 검사하여 새롭게 추가된 파일들을 자동으로 정리해 주는 프로그램

○ 목표

- 새로운 명령어를 시스템 함수를 사용하여 구현함으로써 쉘의 원리를 이해하고, Linux 시스템에서 제공하는 여러 시스템 자료구조와 시스템 콜 및 라이브러리를 이용하여 데몬 프로세스를 생성하여 실행하는 프로그램을 작성함으로써 표준 입출력 및 파일 입출력에 대해 적응하고 이를 응용하여 디렉토리 구조를 링크드 리스트로 구현하며 시스템 프로그래밍 설계 및 응용 능력을 향상

○ 팀 구성

- 개인별 프로젝트

○ 보고서 제출 방법 (강의계획서와 동일. #2 설계과제부터는 보고서 제출 방법은 명세에서 삭제 예정. 강의계획서 참고)

○ 배점 기준 (강의계획서와 동일. #2 설계과제부터는 배점 기준도 명세에서 삭제 예정. 강의계획서 참고)

- 구현 보고서 P설계과제번호.hwp (15점) : 개요 1점, 기능 1점, 상세설계 10점, 실행 결과 3점
- 소스코드 (85점) : 소스코드 주석 5점, 실행 여부 80점 (설계 요구에 따르지 않고 설계된 경우 소스코드 주석 및 실행 여부는 0점 부여. 설계 요구에 따라 설계된 경우 기능 미구현 부분을 설계명세서의 100점 기준에서 해당 기능 감점 후 이를 80점으로 환산)
- 각 설계과제의 완성 유무는 제출 여부로 판단하는 것이 아니라 주어진 과제에서 명시된 "필수구현"의 구현 여부로 판단.
  - ✓ 예. 특정 과제의 필수구현 중 일부만 구현했을 경우 해당 점수는 부여하나 과제는 미구현으로 판단하고 본 교과목 이수조건인 설계과제 최소 구현 개수 1개에 포함시키지 않음
- 구현 보고서 및 소스코드 배점
  - ✓ 보고서는 다음과 같은 양식으로 작성(강의계획서 FAQ 참고) (15점)

1. 과제 개요 (1점) // 명세에 주어진 개요를 더 상세하게 작성
2. 구현 기능 (1점) // 함수 프로토타입 반드시 포함
3. 상세 설계 (10점) // 함수 기능별 흐름도(순서도) 반드시 포함
4. 실행 결과 (3점) // 테스트 프로그램의 실행 결과 캡쳐 및 분석
5. (텍스트 형태의) 소스코드 // 주석 포함, 캡쳐해서 넣지 말고, 텍스트 형태로.

  - ✓ 소스코드 및 실행 여부 (85점) // 주석 (5점), 실행 여부 (80점)

○ ssu_cleanupd 프로그램 기본 사항

- 프로그램 실행 시 내장명령어(show, add, modify, remove, help, exit)에 따라 해당 기능 실행
- ssu_cleanupd 프로그램을 통해 정리 가능한 경로들은 사용자 홈 디렉토리(/home/사용자아이디) 하위 경로여야 함
- ssu_cleanupd 프로그램을 통해 상대경로와 절대경로 모두 입력 가능함
  - ✓ 리눅스 상에서 파일 경로의 최대 크기는 4,096 바이트이며, 파일 이름의 최대 크기는 255 바이트임
- 본 명세에서 특별히 설명하거나 지정하지 않은 것들은 모두 학생들이 스스로 판단하여 구현하면 됨
- 프로그램 전체에서 system() 절대 사용 불가. 사용 시 0점 처리

○ 설계 및 구현

1. ssu_cleanupd

1) Usage : ssu_cleanupd

- ssu_cleanupd 내장명령어는 ssu_cleanupd 프로그램 관련 정보 확인(show), 데몬 프로세스 추가(add), 데몬 프로세스 정보 수정(modify), 데몬 프로세스 등록 해제(remove) 내장명령어 설명(help), 종료(exit)로 이루어져 있음

- 각 내장명령어에 따라 필요한 인자 및 옵션 또한 프로그램 인자를 통해 입력받음

2) 예외 처리

- 프롬프트 상에서 지정한 내장명령어 외 기타 명령어 입력 시 help 명령어의 결과를 출력 후 프롬프트 재출력(3점)
- 프롬프트 상에서 엔터만 입력 시 프롬프트 재출력(2점)

3) 실행 결과

- ssu_cleanupd 실행 시 프롬프트 출력
- ssu_cleanupd의 실행 결과는 각 명령어의 실행 결과를 출력함
- ssu_cleanupd 프로그램 실행 시 사용자의 홈 디렉토리 하위에(~/.ssu_cleanupd/) 디렉토리 및 현재 실행 중인 데몬 프로세스 리스트 파일(~/.ssu_cleanupd/current_deamon_list) 생성
  ✓단, 이미 디렉토리 혹은 파일이 존재하는 경우 생성하지 않음

예제 1. ssu_cleanupd 실행 결과

```
% ./ssu_cleanupd
20230000〉

20230000〉exit

%ls -d ~/.ssu_cleanupd
/home/oslab/.ssu_cleanupd
```

2. 내장명령어 1. show

1) Usage : show

- ssu_cleanupd 프로그램으로 관리하는 데몬 프로세스에 대한 정보를 확인함

2) 실행 결과

- ~/.ssu_cleanupd/current_deamon_list 파일을 읽어 현재 실행 중인 모든 데몬 프로세스의 절대경로를 번호와 함께 출력
- 사용자 입력을 받아 해당 번호에 해당하는 절대경로를 모니터링하는 데몬 프로세스의 상세 정보를 출력
- 선택한 절대경로 하위의 설정 파일(ssu_cleanupd.config) 및 로그파일(ssu_cleanupd.log)을 읽어 아래의 내용들을 출력하며, 출력 형식은 아래 예제 및 add 내장명령어 참조
  ✓ 데몬 프로세스의 옵션 및 PID 등의 정보
  ✓ 최근 10줄(10줄 미만인 경우 전체)의 로그
- 0번(exit) 입력시 show 명령어 종료 후 프롬프트 출력

그림 1-1. /home/oslab/test1 디렉토리 트리 구조

```
/home/oslab/test1
├── A/
├── B/
└── C/
```

예제 2 show 내장명령어 실행(2025-03-20 17:00:00에 데몬 프로세스가 파일 2개를 정리했을 때)

```
% ~/ssu_cleanupd
20230000〉 show
Current working daemon process list

0. exit

Select one to see process info : 0
20230000〉 add /home/oslab/test1/A
20230000〉 add /home/oslab/test1/B
20230000〉 add /home/oslab/test1/C
20230000〉 show
Current working daemon process list

0. exit
1. /home/oslab/test1/A
2. /home/oslab/test1/B
3. /home/oslab/test1/C

Select one to see process info : abc
Please check your input is valid

0. exit
1. /home/oslab/test1/A
2. /home/oslab/test1/B
3. /home/oslab/test1/C

Select one to see process info : 1

1. config detail //cat 명령어 동일한 기능. ssu_cleanupd.config을 보여주는 기능

monitoring_path : /home/oslab/test1/A
pid : 234
strat_time : 2025-02-20 17:00:00
output_path : /home/oslab/test1/A_arranged
time_interval : 10
max_log_lines : none
exclude_path : none
extension : all
mode : 1

2. log detail

[17:00:00] [234] [/home/oslab/cleanup_test/A/1.c] [/home/oslab/cleanup_test_arranged/c/1.c]
[17:00:00] [234] [/home/oslab/cleanup_test/A/2.txt] [/home/oslab/cleanup_test_arranged/txt/2.txt]

20230000〉 exit
%
```

3) 예외 처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)

- 현재 모니터링을 진행하는 데몬 프로세스의 정보를 확인할 때 주어진 번호 이외의 입력이 주어지는 경우 "Please check your input is valid" 출력 후 데몬 프로세스 리스트 재출력 후 다시 번호 입력(2점)

3. 내장명령어 2. add

1) Usage : add 〈DIR_PATH〉 [OPTION] ...

- 모니터링할 디렉토리 경로(DIR_PATH)를 입력받아 해당 디렉토리 내의 파일들 중에 〈DIR_PATH〉_arranged에 정리되지 않은 파일들을 주기적으로 식별하여 정리하는 데몬 프로세스 등록

2) 인자 설명

- 첫 번째 인자 〈DIR_PATH〉는 정리할 디렉토리의 경로이며 상대경로와 절대경로 모두 입력 가능
- 두 번째 인자 [OPTION]은 '-d', '-i', '-l', '-x', '-e', '-m' 가 있으며 모두 생략 가능하고 모두 함께 사용 가능

3) 실행 결과

- 〈DIR_PATH〉_arranged 경로는 〈DIR_PATH〉와 디렉토리 이름을 제외하고 모든 상위 디렉토리 경로가 동일하며, 〈DIR_PATH〉의 디렉토리 이름에 _arranged를 이어 붙인 경로임
- 〈DIR_PATH〉_arranged 디렉토리가 존재하지 않을 경우 생성됨
- 〈DIR_PATH〉_arranged 디렉토리 내에 확장자별 디렉토리(txt, c, cpp 등)가 생성되며, 해당 확장자를 가진 파일들이 각 디렉토리로 복사됨
- 데몬 프로세스 생성을 마쳤다면 프롬프트를 재출력함
- 특정 주기마다 〈DIR_PATH〉 디렉토리와 〈DIR_PATH〉_arranged 디렉토리를 검사하고 정리하는 데몬 프로세스 생성
- add 명령어로 추가되는 데몬 프로세스의 [OPTION] 내용 및 기본적인 프로세스 정보를 〈DIR_PATH〉 하위에 ssu_cleanupd.config 파일을 생성하여 아래 예시와 같은 형식으로 저장하며, 매 주기마다 해당 config 파일을 읽어 설정값을 갱신함
  - ✓ 단, 이미 ssu_cleanupd.config 파일이 존재하는 경우 생성하지 않음
  - ✓ ssu_cleanupd.config 파일은 정리하지 않음
  - ✓ 데몬 프로세스가 주기마다 디렉토리를 검사할 때 config 파일을 읽기 전에 파일 락을 획득하여 modify 명령어를 통해 해당 파일을 접근 중이었다면 대기 후 옵션 갱신

그림 2. ssu_cleanupd.config 파일 형식 (반드시 파일 형식은 지켜야 함, config 인자 : 값은 ":" 으로 구분. 띄어쓰기는 알아서)

```
monitoring_path : 모니터링 되고 있는 디렉토리의 절대 경로
pid : 디렉토리를 모니터링하는 데몬 프로세스의 pid 값
strat_time : 데몬 프로세스가 add된 날짜 및 시간정보 (modify로는 변경 안 됨, remove후 동일 경로 add시 갱신)
output_path : 모니터링하는 파일을 정리할 디렉토리의 절대 경로 (기본값 〈DIR_PATH〉_arranged)
time_interval : 모니터링 주기(초, 기본값 10)
max_log_lines : 로그 파일의 최대 줄 수 (기본값 none)
exclude_path : 제외할 하위 디렉토리의 절대경로, (기본값 none)
extension : 정리할 확장자(기본값 all)
mode : 모드 숫자(기본값 1)
```

- 데몬 프로세스가 정리하여 〈DIR_PATH〉 디렉토리에서 〈DIR_PATH〉_arranged 디렉토리로 복사하는 파일마다 아래 예시와 같은 형식으로 〈DIR_PATH〉 하위의 ssu_cleanupd.log 파일을 생성하여 로그를 작성함
  - ✓ 단, 이미 ssu_cleanupd.log 파일이 존재하는 경우 생성하지 않음
  - ✓ ssu_cleanupd.log 파일은 정리하지 않음
  - ✓ ssu_cleanupd.log에 기록하는 모든 경로는 절대경로임

그림 3. ssu_cleanupd.log 파일 형식

```
[파일 정리 시간] [파일을 정리한 데몬 프로세스의 pid] [정리 전 파일의 절대경로] [정리 후 파일의 절대경로]
[파일 정리 시간] [파일을 정리한 데몬 프로세스의 pid] [정리 전 파일의 절대경로] [정리 후 파일의 절대경로]
                                        ...
[파일 정리 시간] [파일을 정리한 데몬 프로세스의 pid] [정리 전 파일의 절대경로] [정리 후 파일의 절대경로]
```

- 정리 도중 파일명 중복이 발생하는 경우 파일명이 서로 중복되는 파일 중에 mtime을 기준으로 가장 최신 파일을 〈DIR_PATH〉_arranged 디렉토리로 정리함

예제 3 add 명령어 '-d', '-i', '-l' 옵션 실행 결과 (현재 디렉토리가 /home/oslab/test2일 때)

```
% mkdir /home/oslab/test2/A
% mkdir /home/oslab/test2/A_output

% tree /home/oslab/test2
/home/oslab/test2
├── A/
└── A_output/

% ./ssu_cleanupd
20230000〉 add /home/oslab/test2/A -d /home/oslab/test2/A_output -i 15 -l 3          (18:00:00)
20230000〉 exit

% tree /home/oslab/test2
/home/oslab/test2
├── A/
│   ├── ssu_cleanupd.config
│   └── ssu_cleanupd.log
└── A_output/

% cat /home/oslab/test2/ssu_cleanupd.config
monitoring_path : /home/oslab/test2/A
pid : 277
start_time : 2025-03-20 18:00:00
output_path : /home/oslab/test2/A_output
time_interval : 15
max_log_lines : 3
exclude_path : none
extension : all
mode : 1

% touch  /home/oslab/test2/A/1.txt                                    (18:00:05)
% cat /home/oslab/test2/A/ssu_cleanupd.log                            (18:00:20)
[18:00:15] [277] [/home/oslab/test2/A/1.txt] [/home/oslab/test2/A_output/txt/1.txt]

% touch /home/oslab/test2/A/2.c                                       (18:00:27)
% touch /home/oslab/test2/A/3.c                                       (18:00:27)
% touch /home/oslab/test2/A/4.c                                       (18:00:27)

% cat /home/oslab/test2/A/ssu_cleanupd.log                            (18:00:40)
[18:00:30] [277] [/home/oslab/test2/A/2.c] [/home/oslab/test2/A_output/c/2.c]
[18:00:30] [277] [/home/oslab/test2/A/3.c] [/home/oslab/test2/A_output/c/3.c]
[18:00:30] [277] [/home/oslab/test2/A/4.c] [/home/oslab/test2/A_output/c/4.c]
```

- '-d' 옵션 입력 시에는 하나의 〈OUTPUT_PATH〉 옵션 인자를 필요로 하며 정리를 진행할 디렉토리를 〈DIR_PATH〉_arranged 가 아닌 〈OUTPUT_PATH〉 경로의 디렉토리로 설정함
  ✓ 상대경로, 절대경로 모두 입력 가능
- '-i' 옵션 입력 시에는 하나의 〈TIME_INTERVAL〉 옵션 인자를 필요로 하며 데몬 프로세스가 다음 모니터링 할 시간 간격을 〈TIME_INTERVAL〉에 해당하는 초로 설정함
  ✓ 옵션 생략 시 기본값은 10초로 설정
- '-l' 옵션 입력 시에는 하나의 〈MAX_LOG_LINES〉 옵션 인자를 필요로 하며 데몬 프로세스가 기록하는 로그의 최대 줄 수를 〈MAX_LOG_LINES〉에 해당하는 수로 설정함
  ✓ 옵션 생략 시 최대 로그 줄 수는 없으며, ssu_clenaupd.config 파일엔 "none" 문자열로 표기

예제 4 add 명령어 '-x', '-e' 옵션 실행 결과 (현재 디렉토리가 /home/oslab일 때)

```
% mkdir test3
% mkdir test3/A
% mkdir test3/B
% mkdir test3/B/B1
% mkdir test3/C

% tree test3
test3
├── A/
├── B/
│   └── B1/
└── C/

ls -d ../test3_arranged
ls: cannot access 'test3_arranged': No such file or directory

% ./ssu_cleanupd
20230000〉 add test3 -x /home/oslab/test3/B/B1 test3/C -e txt                    (19:00:00)
20230000〉 exit

ls -d ../test3_arranged
../test3_arranged

% tree test3
test3
├── A/
├── B/
│   └── B1/
├── C/
├── ssu_cleanupd.config
└── ssu_cleanupd.log

% cat /home/oslab/test3/ssu_cleanupd.config
monitoring_path : /home/oslab/test3
pid : 303
start_time : 2025-03-20 19:00:00
output_path : /home/oslab/test3_arranged
time_interval : 10
max_log_lines : none
exclude_path : /home/oslab/test3/B/B1,/home/oslab/test3/C
extension : txt
mode : 1

% touch /home/oslab/test3/1.txt                                    (19:00:05)
% touch /home/oslab/test3/A/2.c                                    (19:00:05)
% touch /home/oslab/test3/B/3.txt                                  (19:00:05)
% touch /home/oslab/test3/C/4.c                                    (19:00:06)
% touch /home/oslab/test3/B/B1/5.txt                               (19:00:06)

% cat /home/oslab/test3/ssu_cleanupd.log                           (19:00:15)
[19:00:10] [303] [/home/oslab/test3/1.txt] [/home/oslab/test3_arranged/txt/1.txt]
[19:00:10] [303] [/home/oslab/test3/B/3.txt] [/home/oslab/test3_arranged/txt/3.txt]
```

- '-x' 옵션 입력 시에는 〈EXCLUDE_PATH1, EXCLUDE_PATH2, ...〉 옵션 인자를 필요로 하며 인자로 받은 모

든 디렉토리의 하위 파일들을 제외하고 〈DIR_PATH〉 디렉토리를 정리함
✓ 옵션 생략 시 정리를 제외할 하위 디렉토리는 없으며, ssu_clenaupd.config 파일엔 “none” 문자열로 표기
✓ 인자가 여러 개일 때는 각 하위 디렉토리들을 ‘,’로 구분
✓ 상대경로, 절대경로 모두 입력 가능
- ‘-e’ 옵션 입력 시에는 〈EXTENSION1, EXTENSION2, ...〉 옵션 인자를 필요로 하며 인자로 받은 확장자를 갖는 파일들을 정리함
✓ 옵션 생략 시 지정할 확장자는 없으며, ssu_clenaupd.config 파일엔 “all” 문자열로 표기
✓ 인자가 여러 개일 때는 각 확장자들을 ‘,’로 구분

예제 5 add 명령어 '-m'옵션 실행 결과 (현재 디렉토리가 /home/oslab/test4일 때)

```
% mkdir /A
% mkdir /A/A1
% mkdir /A/A2
% mkdir /B
% mkdir /B/B1
% mkdir /B/B2
% mkdir /C
% mkdir /C/C1
% mkdir /C/C2
% tree test4
test4
├── A/
│   ├── A1/
│   └── A2/
├── B/
│   ├── B1/
│   └── B2/
├── C/
│   ├── C1/
│   └── C2/
└── D/
    ├── d.c
    ├── D1/
    └── D2/

% ls D_arranged
d.c   ssu_cleanupd.config   ssu_cleanupd.log

% ./ssu_cleanupd
20230000〉add A -m 1                                                    (20:00:00)
20230000〉add B -m 2                                                    (20:00:00)
20230000〉add C -m 3                                                    (20:00:00)
20230000〉add D -m 3                                                    (20:00:00)
20230000〉exit

% touch /home/oslab/test4/A/A1/a.c                                      (20:00:01)
% touch /home/oslab/test4/A/A2/a.c                                      (20:00:02)
% touch /home/oslab/test4/B/B1/b.c                                      (20:00:03)
% touch /home/oslab/test4/B/B2/b.c                                      (20:00:04)
% touch /home/oslab/test4/C/C1/c.c                                      (20:00:05)
% touch /home/oslab/test4/C/C2/c.c                                      (20:00:06)
% touch /home/oslab/test4/D/D1/d.c                                      (20:00:07)
% touch /home/oslab/test4/D/D2/d.c                                      (20:00:08)

% cat A/ssu_cleanupd.log B/ssu_cleanupd.log C/ssu_cleanupd.log  D/ssu_cleanupd.log      (20:00:15)
[20:00:10] [443] [/home/oslab/test4/A/A2/a.c] [/home/oslab/test4/A_arranged/c/a.c]
[20:00:10] [444] [/home/oslab/test4/B/B1/b.c] [/home/oslab/test4/B_arranged/c/b.c]
[19:58:50] [440] [/home/oslab/test4/D/d.c] [/home/oslab/test4/D_arranged/c/d.c]
```

- '-m' 옵션 입력 시에는 하나의 〈MODE〉 옵션 인자를 필요로 하며 인자로 받은 MODE 값에 따라 중복된 파일 이름을 처리하는 방식이 달라짐
  - ✓ 〈MODE〉의 값은 1~3 범위를 가짐
  - ✓ 1 : 파일명이 서로 중복되는 파일 중에 mtime을 기준으로 가장 최신 파일을 정리
  - ✓ 2 : 파일명이 서로 중복되는 파일 중에 mtime을 기준으로 가장 오래된 파일을 정리
  - ✓ 3 : 파일명이 서로 중복되는 파일을 정리하지 않음
  - ✓ 옵션 생략 시 기본값은 1으로 설정

4) 예외 처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)
- 첫 번째 인자로 존재하지 않는 경로 혹은 디렉토리 파일이 아니거나 접근권한이 없는 경우 에러 처리 후 프롬프트 재출력(1점)
- 첫 번째 인자로 입력받은 경로가 이미 모니터링 중인 디렉토리거나 입력받은 경로의 하위 혹은 상위 디렉토리인 경우 에러 처리 후 프롬프트 재출력(1점)
- 첫 번째 인자로 입력받은 경로(절대 경로)가 사용자의 홈 디렉토리($HOME, ~)를 벗어나는 경우 "〈입력받은 경로〉 is outside the home directory" 표준 출력 후 프롬프트 재출력(1점)
- 두 번째 인자의 '-d', '-x' 옵션으로 존재하지 않는 경로 혹은 디렉토리 파일이 아니거나 접근권한이 없는 경우 에러 처리 후 프롬프트 재출력(1점)
- 두 번째 인자의 '-d', '-x' 옵션으로 사용자의 홈 디렉토리($HOME, ~)를 벗어나는 경우 혹은 두 번째 인자의 '-d' 옵션으로 첫 번째 인자의 경로를 포함하는 경우에러 처리 후 프롬프트 재출력(1점)
- 두 번째 인자의 '-x' 옵션으로 올바르지 않은 경로(〈DIR_PATH〉의 하위 디렉토리가 아니거나 옵션 인자들 끼리 서로 상위 혹은 하위 디렉토리가 겹치거나 동일한 경로가 존재하는 경우) 경우 에러 처리 후 프롬프트 재출력(1점)
- 두 번째 인자의 '-i', '-l', 옵션으로 자연수가 아닌 입력이 들어왔을 경우 에러 처리 후 프롬프트 재출력(1점)
- 두 번째 인자의 '-m' 옵션으로 범위 밖의 입력이 들어왔을 경우 에러 처리 후 프롬프트 재출력(1점)

4. 내장명령어 3. modify

1) Usage : modify 〈DIR_PATH〉 [OPTION]...
- 모니터링 중인 디렉토리 경로(DIR_PATH)를 입력받아 해당 디렉토리 경로를 모니터링하는 데몬 프로세스의 설정 파일(ssu_cleanupd.config)을 변경하여 해당 설정으로 데몬 프로세스가 동작하게 함

2) 인자 설명
- 내장명령어 2. add와 기본값 제외 동일하며, 입력되지 않은 옵션은 수정하지 않음
- '-d' 옵션 입력 시에는 하나의 〈OUTPUT_PATH〉 옵션 인자를 필요로 하며 정리를 진행할 디렉토리를 〈DIR_PATH〉_arranged 가 아닌 〈OUTPUT_PATH〉 경로의 디렉토리로 설정함
  - ✓ 상대경로, 절대경로 모두 입력 가능
- '-i' 옵션 입력 시에는 하나의 〈TIME_INTERVAL〉 옵션 인자를 필요로 하며 데몬 프로세스가 다음 모니터링할 시간 간격을 〈TIME_INTERVAL〉에 해당하는 초로 설정함
- '-l' 옵션 입력 시에는 하나의 〈MAX_LOG_LINES〉 옵션 인자를 필요로 하며 데몬 프로세스가 기록하는 로그의 최대 줄 수를 〈MAX_LOG_LINES〉에 해당하는 수로 설정함
- '-x' 옵션 입력 시에는 〈EXCLUDE_PATH1, EXCLUDE_PATH2, ...〉 옵션 인자를 필요로 하며 인자로 받은 모든 디렉토리의 하위 파일들을 제외하고 〈DIR_PATH〉 디렉토리를 정리함
  - ✓ 인자가 여러 개일 때는 각 하위 디렉토리들을 ','로 구분
  - ✓ 상대경로, 절대경로 모두 입력 가능
- '-e' 옵션 입력 시에는 〈EXTENSION1, EXTENSION2, ...〉 옵션 인자를 필요로 하며 인자로 받은 확장자를 갖는 파일들을 정리함
  - ✓ 인자가 여러 개일 때는 각 확장자들을 ','로 구분
- '-m' 옵션 입력 시에는 하나의 〈MODE〉 옵션 인자를 필요로 하며 인자로 받은 MODE 값에 따라 중복된 파일 이름을 처리하는 방식이 달라짐
  - ✓ 〈MODE〉의 값은 1~3 범위를 가짐
  - ✓ 1 : 파일명이 서로 중복되는 파일 중에 mtime을 기준으로 가장 최신 파일을 정리
  - ✓ 2 : 파일명이 서로 중복되는 파일 중에 mtime을 기준으로 가장 오래된 파일을 정리
  - ✓ 3 : 파일명이 서로 중복되는 파일을 정리하지 않음

예제 6 modify 명령어 실행 결과 (현재 디렉토리가 /home/oslab/일 때)

```
% mkdir test5

% ./ssu_cleanupd
20230000〉 add test5 -i 5 -l 5                                    (21:00:00)
20230000〉 exit

% cat test5/ssu_cleanup.config
monitoring_path : /home/oslab/test5
pid : 777
strat_time : 2025-03-20 21:00:00
output_path : /home/oslab/test5_arranged
time_interval : 5
max_log_lines : 5
exclude_path : none
extension : all
mode : 1

% touch /home/oslab/test5/a.c                                    (21:00:03)

% ./ssu_cleanupd
20230000〉 modify test5 -i 10 -e txt                             (21:00:04)
20230000〉 exit

% cat test5/ssu_cleanup.config
monitoring_path : /home/oslab/test5
pid : 777
strat_time : 2025-03-20 21:00:00
output_path : /home/oslab/test5_arranged
time_interval : 5
max_log_lines : 5
exclude_path : none
extension : txt
mode : 1

% touch /home/oslab/test5/b.c                                    (21:00:07)

% cat test5/ssu_cleanupd.log                                     (20:00:17)
[21:00:05] [777] [/home/oslab/test5/a.c] [/home/oslab/test5_arranged/c/a.c]
[21:00:15] [777] [/home/oslab/test5/b.c] [/home/oslab/test5_arranged/c/b.c]
```

3) 실행 결과

- 〈DIR_PATH〉 하위의 ssu_cleanupd.config 파일의 내용을 입력한 옵션 인자 값으로 설정함

✓ config 파일에 설정한 인자 값을 쓰기 전에 파일 락을 획득하여 모니터링 중인 데몬 프로세스가 옵션 값을 읽기 위해 해당 파일을 접근 중이었다면 대기 후 옵션 갱신

4) 예외 처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)

- 첫 번째 인자에 대한 예외 처리, 모두 구현해야 점수(1점)

✓ 첫 번째 인자로 존재하지 않는 경로 혹은 디렉토리 파일이 아니거나 접근권한이 없는 경우 에러 처리 후 프롬프트 재출력

✓ 첫 번째 인자로 입력받은 경로가 이미 모니터링 중인 디렉토리가 아닌 경우 에러 처리 후 프롬프트 재출력

✓ 첫 번째 인자로 입력받은 경로(절대 경로)가 사용자의 홈 디렉토리($HOME, ~)를 벗어나는 경우 "〈입력받은 경로〉 is outside the home directory" 표준 출력 후 프롬프트 재출력

- 두 번째 인자 중 '-d', '-x' 옵션에 대한 예외 처리, 모두 구현해야 점수(1점)

✓ 두 번째 인자의 '-d', '-x' 옵션으로 존재하지 않는 경로 혹은 디렉토리 파일이 아니거나 접근권한이 없는 경우 에러 처리 후 프롬프트 재출력

✓ 두 번째 인자의 '-d', '-x' 옵션으로 사용자의 홈 디렉토리($HOME, ~)를 벗어나는 경우 에러 처리 후 프롬프

트 재출력

✓ 두 번째 인자의 '-x' 옵션으로 올바르지 않은 경로(〈DIR_PATH〉의 하위 디렉토리가 아니거나 옵션 인자들끼리 서로 상위 혹은 하위 디렉토리가 겹치거나 동일한 경로가 존재하는 경우) 경우 에러 처리 후 프롬프트 재출력

✓ 두 번째 인자의 '-d' 옵션으로 첫 번째 인자의 경로를 포함하는 경우 에러 처리 후 프롬프트 재출력

- 두 번째 인자 중 '-i', '-l', '-m' 옵션에 대한 예외 처리, 모두 구현해야 점수(1점)

✓ 두 번째 인자의 '-i', '-l' 옵션으로 자연수가 아닌 입력이 들어왔을 경우 에러 처리 후 프롬프트 재출력

✓ 두 번째 인자의 '-m' 옵션으로 범위 밖의 입력이 들어왔을 경우 에러 처리 후 프롬프트 재출력

5. 내장명령어 4. remove

1) Usage : remove 〈DIR_PATH〉

- 모니터링 중인 디렉토리 경로(DIR_PATH)를 입력받아 해당 디렉토리를 모니터링 하고 있는 프로세스를 종료하는 명령어

2) 실행 결과

- 〈DIR_PATH〉 경로를 모니터링 하는 데몬 프로세스가 종료됨

예제 7 modify 명령어 실행 결과 (현재 디렉토리가 /home/oslab/일 때)

```
% mkdir test6
% mkdir test6/A
% mkdir test6/B

% ./ssu_cleanupd
20230000〉 add test6/A
20230000〉 add test6/B
20230000〉 show
Current working daemon process list

0. exit
1. /home/oslab/test6/A
2. /home/oslab/test6/B

Select one to see process info : 0
20230000〉 remove test6/A
20230000〉 show
Current working daemon process list

0. exit
1. /home/oslab/test6/B

Select one to see process info : 0
20230000〉 exit
```

3) 예외 처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)

- 첫 번째 인자로 존재하지 않는 경로 혹은 디렉토리 파일이 아니거나 접근권한이 없는 경우 에러 처리 후 프롬프트 재출력(1점)
- 첫 번째 인자로 입력받은 경로가 이미 모니터링 중인 디렉토리가 아닌 경우 에러 처리 후 프롬프트 재출력(1점)
- 첫 번째 인자로 입력받은 경로(절대 경로)가 사용자의 홈 디렉토리($HOME, ~)를 벗어나는 경우 "〈입력받은 경로〉 is outside the home directory" 표준 출력 후 프롬프트 재출력(1점)

6. 내장명령어 5. help

1) Usage : help

- 프로그램 내장명령어에 대한 설명(Usage) 출력

2) 실행 결과

- 프로그램 내장명령어 help 실행

예제 8. help 내장명령어 실행

```
20230000〉 help
Usage:
  〉 show
    〈none〉 : show monitoring deamon process info
  〉 add 〈DIR_PATH〉 [OPTION]...
    〈none〉 : add deamon process monitoring the 〈DIR_PATH〉 directory
    -d 〈OUTPUT_PATH〉 : Specify the output directory 〈OUTPUT_PATH〉 where 〈DIR_PATH〉 will be
                        arranged
    -i 〈TIME_INTERVAL〉 : Set the time interval for the daemon process to monitor in seconds.
    -l 〈MAX_LOG_LINES〉 : Set the maximum number of log lines the daemon process will record.
    -x 〈EXCLUDE_PATH1, EXCLUDE_PATH2, ...〉 : Exclude all subfiles in the specified directories.
    -e 〈EXTENSION1, EXTENSION2, ...〉 : Specify the file extensions to be organized.
    -m 〈M〉 : Specify the value for the 〈M〉 option.
  〉 modify 〈DIR_PATH〉 [OPTION]...
    〈none〉 : modify deamon process config monitoring the 〈DIR_PATH〉 directory
    -d 〈OUTPUT_PATH〉 : Specify the output directory 〈OUTPUT_PATH〉 where 〈DIR_PATH〉 will be
                        arranged
    -i 〈TIME_INTERVAL〉 : Set the time interval for the daemon process to monitor in seconds.
    -l 〈MAX_LOG_LINES〉 : Set the maximum number of log lines the daemon process will record.
    -x 〈EXCLUDE_PATH1, EXCLUDE_PATH2, ...〉 : Exclude all subfiles in the specified directories.
    -e 〈EXTENSION1, EXTENSION2, ...〉 : Specify the file extensions to be organized.
    -m 〈M〉 : Specify the value for the 〈M〉 option.
  〉 remove 〈DIR_PATH〉
    〈none〉 : remove deamon process monitoring the 〈DIR_PATH〉 directory
  〉 help
  〉 exit
```

7. 내장명령어 6. exit

1) Usage : exit

- 현재 실행중인 ssu_cleanupd 프로그램 종료

2) 실행 결과

- 프로그램 종료

예 9. exit 실행 시 프로그램 종료

```
20230000〉 exit
%
```

○ 과제 구현에 필요한 함수 (필수 아님)

- 1. getopt() : 프로그램 실행 시 입력한 인자를 처리하는 라이브러리 함수

```
#include 〈unistd.h〉
int getopt(int argc, char * const argv[], const char *optstring); //_POSIX_C_SOURCE

#include 〈getopt.h〉
int getopt_long(int argc, char * const argv[], const char *optstring, const struct option *longopts, int *longindex);
//_GNU_SOURCE
```

- 2. scandir : 디렉토리에 존재하는 파일 및 디렉토리 전체 목록 조회하는 라이브러리 함수

```
#include 〈dirent.h〉
int scandir(const char *dirp, struct dirent ***namelist, int (*filter)(const struct dirent *), int (*compar)(const struct
dirent **, const struct dirent **));

-1 : 에러가 발생, 상세한 에러 내용은 errno에 설정
0 이상 : 정상적으로 처리, namelist에 저장된 struct dirent *의 개수가 return
```

- 3. realpath : 상대경로를 절대경로로 변환하는 라이브러리 함수

```
#include <stdlib.h>
char *realpath(const char *path, char *resolved_path);

NULL : 에러가 발생, 상세한 에러 내용은 errno 전역변수에 설정
NULL이 아닌 경우 : resolved_path가 NULL이 아니면, resolved_path를 return,
resolved_path가 NULL이면, malloc(3)으로 할당하여 real path를 저장한 후에 return
```

- 4. strtok : 특정 문자 기준으로 문자열을 분리하는 함수

```
#include <string.h>
char *strtok(char *restrict str, const char *restrict delim);

 return a pointer to the next token, or NULL if there are no more tokens.
```

- 5. exec()류 함수 : 현재 프로세스 이미지를 새로운 프로세스 이미지로 대체하는 라이브러리/시스템콜 함수

```
#include <unistd.h>
int execl(const char *pathname, const char *arg, .../* (char  *) NULL */);
int execv(const char *pathname, char *const argv[]);
int execle(const char *pathname, const char *arg, .../*, (char *) NULL, char *const envp[] */);
int execve(const har* pathname, char *const argv[], char *const ecnp[]); //시스템콜
int execlp(const char *file, const char *arg, .../* (char  *) NULL */);
int execvp(const char *file, char *const argv[]);
int execvpe(const char *file, char *const argv[],char *const envp[]);

 The exec() family of functions replaces the current process image with a new process image.
 https://man7.org/linux/man-pages/man3/exec.3.html  또는 교재 참고
```

- 6. fcntl : add가 등록하는 데몬 프로세스 및 modify 명령어에서 ssu_cleand.config 파일 접근 시 필요

```
#include <unistd.h>
#include <fnctl.h>

int fcntl(int fd, int cmd, ... /* arg */);

int fcntl(int fd, int cmd);
int fcntl(int fd, int cmd, long arg);
int fcntl(int fd, int cmd, struct flock *lock); // 본 프로그램에서는 cmd 값으로 F_SETLKW 플래그를 사용하는 것을 추천


 fcntl() performs one of the operations described below on the open file descriptor fd.  The operation is determined by op.
 https://man7.org/linux/man-pages/man2/fcntl.2.html  또는 교재 455 페이지 함수 설명 및 481 페이지 예제 참고
```

✓ 아래 예제 프로그램을 컴파일하여 다른 두 개의 터미널에서 두 가지 경우를 실험하는 것을 추천

✓ 1. 1번 터미널에서 ./fcntl_test 2 실행

✓ 2. 1번 터미널에서 ./fcntl_test 1 실행 후 2번 터미널에서 fcntl_test 2 실행 후 1번 터미널에 아무 글자 입력

예 10. fcntl 예제 프로그램 (fcntl_test.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
int main(int argc, char *argv[]){
    int fd;
    char c, *fname = "input.txt";
    struct flock lock;
    if(argc < 2) {
        printf("Usage : %s (1/0)₩n", argv[0]);
        exit(1);
    }

    if((fd = open(fname, O_RDWR)) < 0){
        printf("open error for %s₩n", fname);
        exit(1);
    }

    lock.l_type = F_WRLCK;
    lock.l_whence = SEEK_SET;
    lock.l_start = 0;
    lock.l_len = 0;

    if(fcntl(fd, F_SETLKW, &lock) == -1){
        printf("fcntl error for %s₩n", fname);
        close(fd);
        exit(1);
    }

    if(argv[1][0] == '1')
        while((c = getchar()) == '₩n'); //글자 입력 대기
    else
        printf("get lock!₩n");

    lock.l_type = F_UNLCK;

    if(fcntl(fd, F_SETLKW, &lock) == -1){
        printf("fcntl error for %s₩n", fname);
        close(fd);
        exit(1);
    }
    close(fd);
}
```

○ 보고서 제출 시 유의 사항

- 보고서 제출 마감은 **4월 29일(화) 오후 11시 59분 59초**

※ 구글클래스룸에서 11:59:00에서 1초라도 지연되면 1일 지연제출로 나올 것임. 이 부분은 추후 제출 시간 확인 후 11:59:59 이내인 경우 정상 제출로 인정 예정임. 단, 1초 지연제출도 0점 처리

- 압축 에러, 파일 누락 관련 감점 syllabus 참고

- 필수구현 : 1. ssu_cleanupd, 2. 내장명령어 show, add, modify, remove help, exit(예외처리는 필수구현 아님. 단, 별도 감점 있음)

- 배점(100점 만점. 실행 여부 배점 80점으로 최종 환산하며, 보고서 15점과 소스코드 주석 5점은 별도, 강의계획

서 확인)

1. ssu_cleanupd : 5점 (필수)
2. 내장명령어 1. show – 15점 (필수)
3. 내장명령어 2. add – 50점, 아래 옵션 미구현 시 : 38점
   - ✓ '-d' 옵션 : 2점
   - ✓ '-l' 옵션 : 2점 (필수)
   - ✓ '-i' 옵션 : 2점 (필수)
   - ✓ '-x' 옵션 : 2점
   - ✓ '-e' 옵션 : 2점
   - ✓ '-m' 옵션 : 2점
4. 내장명령어 3. modify – 10점, 아래 옵션 미구현 시 : 7점
   - ✓ '-d' 옵션 : 0.5점
   - ✓ '-l' 옵션 : 0.5점 (필수)
   - ✓ '-i' 옵션 : 0.5점 (필수)
   - ✓ '-x' 옵션 : 0.5점
   - ✓ '-e' 옵션 : 0.5점
   - ✓ '-m' 옵션 : 0.5점
5. 내장명령어 4. remove – 5점 (필수)
6. 내장명령어 5. help – 5점 (필수)
7. 내장명령어 6. exit – 5점 (필수)
8. makefile 작성(매크로 사용하지 않아도 됨) – 5점 (필수)

※ (가산점 부여기준 1) 4월 15일(화) 밤 11시59분까지 명세서에서 언급한 모든 기능(필수구현 외 기타 구현 모두)을 구현하고 보고서를 제출한 경우 본과제 총점에 +30점, 4월 22일(화) 밤 11시59분까지 제출한 경우 본과제 총점(100점)에 +10점을 (가산점으로)추가 부여함.