# 설계과제 3 개요 : SSU-Sync

Linux System Programming, School of CSE, Soongsil University, Spring 2024

○ 개요
- 리눅스 시스템 상에서 사용자가 지정한 경로에 대해 해당 경로 및 하위 경로들에 대해 데몬 프로세스로 모니터링하고, 내용이 달라졌을 경우 백업 및 이를 로그로 관리하는 프로그램

○ 목표
- 새로운 명령어를 시스템 함수를 사용하여 구현함으로써 쉘의 원리를 이해하고, 유닉스/리눅스 시스템에서 제공하는 여러 시스템 자료구조와 시스템 콜 및 라이브러리를 이용하여 데몬 프로세스를 생성하여 실행하는 프로그램을 작성함으로써 백그라운드 프로세스에 대한 이해를 높이고 이를 응용하여 디렉토리 구조를 링크드 리스트로 구현하며 시스템 프로그래밍 설계 및 응용 능력을 향상

○ 팀 구성
- 개인별 프로젝트

○ 보고서 제출 방법
- 설계과제는 “#P설계과제번호_학번.zip” (예. #P1_20140000_V1.zip)형태로 압축하여 classroom.google.com에 제출해야 함.
- “#P설계과제번호_학번.zip” 내 보고서인 “#P설계과제번호.hwp” 와 헤더파일 및 소스코드 등 해당 디렉토리에서 컴파일과 실행이 가능하도록 모든 파일(makefile, obj, *.c, *.h 등 컴파일하고 실행하기 위한 파일들)을 포함시켜야 함. 단, 특정한 디렉토리에서 실행해야 할 경우는 예외.
- 구현보고서인 “#P설계과제번호.hwp”에는 1. 과제개요(명세에서 주어진 개요를 그대로 쓰면 안됨. 자기가 구현한 내용 요약) 2. 기능(구현한 기능 요약), 3. 상세설계(함수 및 모듈 구성, 순서도, 구현한 함수 프로토타입 등), 4. 실행결과(구현한 모든 기능 및 실행 결과 캡쳐)를 반드시 포함시켜야 함.
- 제출한 압축 파일을 풀었을 때 해당 디렉토리에서 컴파일 및 실행이 되어야 함(특정한 디렉토리에서 실행해야 하는 경우는 제외). 해당 디렉토리에서 컴파일이나 실행이 되지 않을 경우, 혹은 기본과제 및 설계과제 제출 방법(파일명, 디렉토리명, 컴파일 시에 포함되어야 할 파일 등)을 따르지 않으면 무조건 해당 과제 배점의 50% 감점
- 설계과제를 기한 내 새로 제출할 경우 기존 것은 삭제하지 않고 #P설계과제번호_학번_V1.zip 형태로 버전 번호를 붙이면 됨. 버전 이름은 대문자 V와 함께 integer를 1부터 incremental 증가시키면서 부여 (예. #P3_20140000_V1.zip, #P3_20140000_V2.zip) 하면 됨. 단, 처음 제출 시는 버전 번호를 붙이지 않아도 되며 두 번째부터 V1를 붙여 제출하면 됨. 기한 내에 여러 버전의 보고서를 제출할 수 있으나, 채점은 최종 버전만을 대상으로 함.

✓ 설계과제명세서와 강의계획서 상 배점 기준이 다를 경우 해당 설계과제명세서의 배점 기준이 우선 적용

✓ 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 “필수기능요건” 의 구현으로 판단. (예. 특정 과제의 필수 기능 중 일부 기능만 구현했을 경우 해당 점수는 부여하나 과제는 미구현으로 판단하고 본 교과목 이수조건인 설계 과제 최소 구현 개수 2개에 포함시키지 않음)

- 기타 내용은 강의계획서 참고

○ 제출 기한
- 5월 26일(일) 오후 11시 59분 59초

○ 보고서 및 소스코드 배점
- 보고서는 다음과 같은 양식으로 작성(강의계획서 FAQ 참고)

> 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. 실행결과 (3점) // 테스트 프로그램의 실행결과 캡쳐 및 분석

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

○ ssu_sync 프로그램 기본 사항
- 리눅스 시스템에서 생성한 자신의 사용자 아이디에 대해 사용자 홈 디렉토리를 확인 (예. 자신의 리눅스 시스템상 계정 아이디가 oslab일 경우 $HOME은 /home/oslab임. (%echo $HOME, 또는 %cat /etc/passwd 파일에서 해당 아이디의 홈

디렉토리 확인)

- ssu_sync 프로그램의 백업 경로는 기본적으로 ~/backup이며 프로그램 실행 시 사용자의 홈 디렉토리 내에 /backup 디렉토리가 없다면 생성하고, 백업 디렉토리 내에 로그 파일(monitor_list.log)이 없다면 생성
- ssu_sync 프로그램을 통해 입력 가능한 경로들은 사용자 홈 디렉토리(/home/사용자아이디) 내 경로여야 하며, 상대경로와 절대경로 모두 입력 가능함
  ✓ 리눅스 상에서 파일 경로의 최대 크기는 4,096 바이트이며, 파일 이름의 최대 크기는 255 바이트임
- ssu_sync 프로그램 실행 시 내장명령어(add, remove, list, help, exit)에 따라 해당 기능 실행
- ssu_sync 프로그램은 모니터링 대상 디렉토리를 지정하고, 지정한 디렉토리 내의 모든 정규 파일의 변경 상태를 모니터링 하며 대상 디렉토리 하위의 모든 정규 파일을 모니터링하는 데몬 프로세스 생성
- ssu_sync 프로그램을 통해 데몬 프로세스 생성 시 해당 프로세스가 추적중인 디렉토리에 대해 백업 디렉토리 내 해당 데모 프로세스의 pid를 이름으로 하는 디렉토리 생성 및 백업 디렉토리 내 로그파일(monitor_list.log)에 기록
- 데몬 프로세스는 파일이 생성, 삭제, 수정된 경우 모니터링 백업 디렉토리 내 해당 데몬 프로세스 pid를 이름으로 하는 로그 파일("PID".log) 파일에 변경 사항 기록

예. 데몬 프로세스가 관리하는 로그 파일 형태(현재 작업 디렉토리(pwd)가 "/home/oslab"일 경우)

```
% ./ssu_symc
20220000〉 add ./testdir -r
monitoring started (/home/oslab/testdir) : 5230

20220000〉 exit
% cat /home/oslab/backup/monitor_list.log
5230 : /home/oslab/testdir

% cat ~/backup/5230.log
[2024-05-03 10:00:00][create][/home/oslab/testdir/a.txt]
[2024-05-03 10:00:15][create][/home/oslab/testdir/b.txt]
[2024-05-03 10:00:30][create][/home/oslab/testdir/a/a.txt]
[2024-05-03 10:01:05][create][/home/oslab/testdir/c.txt]
```

  ✓ [수행 시간] [수행 내용] [파일의 절대경로] 형태로 작성
  ✓ 파일 변경 사항은 pid 로그 파일의 끝에 추가
- 데몬 프로세스를 통해 파일의 변경을 탐지하기 위해 비교할 경우 stat 구조체를 통해 사이즈 및 mtime, ctime을 비교하며 내용에 대한 비교는 md5를 이용하여 해시값을 통해 비교
- 프로그램 전체에서 system() 절대 사용 불가. 사용 시 0점 처리
- getopt() 라이브러리를 사용하여 옵션 처리 권장
- ssu_sync 프로그램 자체는 foreground로만 수행되는 것으로 가정함(& background 수행은 안되는 것으로 함)

○ 설계 및 구현

**1. ssu_sync**

1) Usage : ssu_sync
   - ssu_sync 프로그램 실행 시 사용자 입력을 기다리는 프롬프트 출력
2) 인자 설명
   - 프로그램 실행시 인자는 따로 받지 않음
3) 실행결과
   - ssu_sync 프로그램의 실행결과는 "학번〉"이 표준 출력되고 내장명령어(add, remove, list, help, exit) 입력 대기. 학번이 20220000일 경우 "20220000〉" 형태로 출력
   - ssu_sync 프로그램을 실행한 사용자의 홈 디렉토리 내에 backup 디렉토리가 없을 경우 생성하고, 내부에 monitor_list.log 파일이 없을 경우 생성. 이미 존재할 경우에는 생성하지 않음
   - ssu_sync 프로그램을 통해 입력받은 경로는 백업 디렉토리를 포함하지 않으며 사용자 홈 디렉토리 내의 경로(사용자 홈 디렉토리 제외)여야 함

예제 1. ssu_sync 실행결과

```
% ./ssu_sync
20220000〉
20220000〉 asd
Usage:
  〉 add 〈PATH〉 [OPTION]... : add new daemon process of 〈PATH〉 if 〈PATH〉 is file
     -d : add new daemon process of 〈PATH〉 if 〈PATH〉 is directory
     -r : add new daemon process of 〈PATH〉 recursive if 〈PATH〉 is directory
     -t 〈PERIOD〉 : set daemon process time to 〈PERIOD〉 sec (default : 1sec)
  〉 remove 〈DAEMON_PID〉 : delete daemon process with 〈DAEMON_PID〉
  〉 list [DAEMON_PID] : show daemon process list or dir tree
  〉 help [COMMAND] : show commands for program
  〉 exit : exit program

% ./ssu_sync
20220000〉 exit

%
```

4) 예외처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)
- 프롬프트 상에서 엔터만 입력 시 프롬프트 재출력
- 프롬프트 상에서 지정한 내장명령어 외 기타 명령어 입력 시 help 명령어의 결과를 출력 후 프롬프트 재출력

**2. 내장명령어 1. add**

1) Usage : add 〈PATH〉 [OPTION]...
- 백그라운드에서 모니터링하며 변경내역을 추적하여 백업할 경로(PATH)를 입력받아 해당 파일 또는 디렉토리에 대한 데몬 프로세스를 추가

2) 인자 설명
- 첫 번째 인자 〈PATH〉는 스테이징 구역에 추가할 파일이나 디렉토리의 상대경로와 절대경로 모두 입력 가능해야 함
- 두 번째 인자 [OPTION]은 '-d', '-r', '-t'가 있으며 동시 사용 가능하고, 생략 가능함('-d', '-r', '-t' 옵션은 아래 설명 확인)

3) 실행결과
- 첫 번째 인자 〈PATH〉에 대해 변경내역을 추적하며 변경되었을 시 백업을 진행하는 데몬 프로세스를 생성하여 〈PATH〉의 모니터링 시작
- 데몬 프로세스는 〈PATH〉를 1초 주기로 모니터링함
- 백업 디렉토리 내 "monitor_list.log"에 〈PATH〉의 절대경로와 생성된 데몬 프로세스의 pid 정보 저장
- 백업 디렉토리 내 데몬 프로세스 pid를 이름으로 하는 디렉토리 생성 및 해당 디렉토리 내에 (원본 파일 이름_수행 시간)을 이름으로 하는 백업 파일 생성. 삭제의 경우 따로 백업 파일을 생성하지 않음
- "monitor_list.log"에 존재하는 데몬 프로세스를 외부에서 kill 등으로 종료하는 경우는 고려하지 않음

예제 2-1. add 내장명령어 실행결과(현재 작업 디렉토리(pwd)가 "/home/oslab"일 경우)

```
% ./ssu_sync
20220000〉 add testdir
"/home/oslab/testdit" is directory

20220000〉 add a.txt
monitoring started (/home/oslab/a.txt) : 5230

20220000〉 exit
% cat /home/oslab/backup/monitor_list.log
5230 : /home/oslab/P3/testdir
% cat /home/oslab/backup/5230.log
[2024-05-03 10:00:00][create][/home/oslab/a.txt]
% vi a.txt
% rm a.txt
% cat /home/oslab/backup/5230.log
[2024-05-03 10:00:00][create][/home/oslab/a.txt]
[2024-05-03 10:00:30][modify][/home/oslab/a.txt]
[2024-05-03 10:00:43][remove][/home/oslab/a.txt]
```

예제 2-1-1. /home/oslab/backup 디렉토리 트리 구조

```
/home/oslab/backup
├ 5230/
|  ├ a.txt_20240503100000
|  └ a.txt_20240503100030
├ 5230.log
└ monitor_list.log
```

- '-d' 옵션 입력시에는 〈PATH〉가 디렉토리일 시 해당 경로 아래 있는 모든 파일들에 대해 모니터링 하는 데몬 프로세스를 생성함.

예제 2-2. add 내장명령어 '-d' 옵션 실행결과(현재 작업 디렉토리(pwd)가 "/home/oslab"일 경우)

```
% ./ssu_sync
20220000〉 add testdir -d
monitoring started (/home/oslab/testdir) : 5230

20220000〉 exit
% cat /home/oslab/backup/monitor_list.log
5230 : /home/oslab/P3/testdir
% cat /home/oslab/backup/5230.log
[2024-05-03 10:00:00][create][/home/oslab/testdir/a.txt]
[2024-05-03 10:00:00][create][/home/oslab/testdir/b.txt]
[2024-05-03 10:00:00][create][/home/oslab/testdir/c.txt]
% vi testdir/a.txt
% rm ./testdir/b.txt
% touch testdir/d.txt
% touch testdir/a/b/a.txt
% cat /home/oslab/backup/5230.log
[2024-05-03 10:00:00][create][/home/oslab/testdir/a.txt]
[2024-05-03 10:00:00][create][/home/oslab/testdir/b.txt]
[2024-05-03 10:00:00][create][/home/oslab/testdir/c.txt]
[2024-05-03 10:00:30][modify][/home/oslab/testdir/a.txt]
[2024-05-03 10:00:43][remove][/home/oslab/testdir/b.txt]
[2024-05-03 10:01:24][create][/home/oslab/testdir/d.txt]
```

예제 2-2-1. /home/oslab/backup 디렉토리 트리 구조

```
/home/oslab/backup
├ 5230/
|  ├ a.txt_20240503100000
|  ├ a.txt_20240503100030
|  ├ b.txt_20240503100000
|  ├ c.txt_20240503100000
|  └ d.txt_20240503100124
├ 5230.log
└ monitor_list.log
```

- '-r' 옵션 입력시에는 〈PATH〉가 디렉토리일 시 해당 경로 아래 있는 모든 파일들에 대해 모니터링 하는 데몬 프로세스를 생성함. 이 때, 서브 디렉토리 내에 모든 파일들에 대해서도 재귀적으로 탐색하여 모니터링을 진행함. 진행 순서는 파일을 우선으로 하고 그 다음 서브디렉토리를 재귀적으로 탐색함(BFS)
- '-r' 옵션과 '-d' 옵션을 동시 입력 시에는 '-r' 옵션을 적용시켜 재귀적으로 원본 파일에 대한 모니터링 하는 데몬 프로세스를 생성함.

예제 2-3. add 내장명령어 '-d' 옵션 실행결과(현재 작업 디렉토리(pwd)가 "/home/oslab"일 경우)

```
% ./ssu_sync
20220000〉 add testdir -r
monitoring started (/home/oslab/testdir) : 5230

20220000〉 exit
% cat /home/oslab/backup/monitor_list.log
5230 : /home/oslab/P3/testdir
% cat /home/oslab/backup/5230.log
[2024-05-03 10:00:00][create][/home/oslab/testdir/a/a.txt]
[2024-05-03 10:00:00][create][/home/oslab/testdir/a.txt]
[2024-05-03 10:00:00][create][/home/oslab/testdir/b.txt]
[2024-05-03 10:00:00][create][/home/oslab/testdir/c.txt]
% vi testdir/a.txt
% rm ./testdir/b.txt
% touch testdir/d.txt
% touch testdir/a/b/a.txt
% cat /home/oslab/backup/5230.log
[2024-05-03 10:00:00][create][/home/oslab/testdir/a/a.txt]
[2024-05-03 10:00:00][create][/home/oslab/testdir/a.txt]
[2024-05-03 10:00:00][create][/home/oslab/testdir/b.txt]
[2024-05-03 10:00:00][create][/home/oslab/testdir/c.txt]
[2024-05-03 10:00:30][modify][/home/oslab/testdir/a.txt]
[2024-05-03 10:00:43][remove][/home/oslab/testdir/b.txt]
[2024-05-03 10:01:07][create][/home/oslab/testdir/a/b/a.txt]
[2024-05-03 10:01:24][create][/home/oslab/testdir/d.txt]
```

예제 2-3-1. /home/oslab/backup 디렉토리 트리 구조

```
/home/oslab/backup
├ 5230/
|  ├ a/
|  |  ├ a.txt_20240503100000
|  |  └ b/
|  |     └ a.txt_20240503100107
|  ├ a.txt_20240503100000
|  ├ a.txt_20240503100030
|  ├ b.txt_20240503100000
|  ├ c.txt_20240503100000
|  └ d.txt_20240503100124
├ 5230.log
└ monitor_list.log
```

- '-t' 옵션 입력 시에는 〈PERIOD〉를 입력받아 해당 주기(초)마다 데몬 프로세스가 실행될 수 있도록 함

4) 예외 처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)
- 경로를 입력하지 않을 경우, add 명령어에 대한 에러 처리 및 Usage 출력 후 프롬프트 재출력
- 인자로 입력받은 경로가 길이 제한(4,096 Byte)를 넘거나 경로가 올바르지 않은 경우, 에러 처리 후 프롬프트 재출력
- 인자로 입력받은 경로가 사용자 홈 디렉토리 내의 경로(사용자 홈 디렉토리 제외)가 아니거나 백업 디렉토리를 포함하는 경로일 경우, 에러 처리 후 프롬프트 재출력
- 인자로 입력받은 옵션 중 -d, -r을 사용하지 않았는데, 인자로 입력받은 경로가 디렉토리일 경우, 에러 처리 후 프롬프트 재출력
- 인자로 입력받은 옵션 중 -d, -r을 사용하였는데, 인자로 입력받은 경로가 파일일 경우, 에러 처리 후 프롬프트 재출력

- 인자로 입력받은 경로에 대해 데몬 프로세스가 이미 존재한다면 에러 처리 후 프롬프트 재출력
- '-t' 옵션의 사용이 올바르지 않은 경우 Usage 출력 후 프롬프트 재출력

### 3. 내장명령어 2. remove

1) Usage : remove 〈DAEMON_PID〉
- pid를 이용해 데몬 프로세스를 제거함으로써 모니터링을 중지하고 백업 파일을 삭제함

2) 인자 설명
- 첫 번째 인자 〈DAEMON_PID〉는 현재 모니터링을 진행 중인 데몬 프로세스의 pid 중 하나

3) 실행결과
- 첫 번째 인자 〈DAEMON_PID〉에 SIGUSR1 시그널을 보내 데몬 프로세스를 종료함
- "monitor_list.log"에서 해당 데몬 프로세스 정보 삭제 및 해당 데몬 프로세스의 로그 파일 및 디렉토리 삭제

예제 3. remove 내장명령어 실행결과(현재 작업 디렉토리(pwd)가 "/home/oslab"일 경우)

```
% cat ~/backup/monitor_list.log
5230 : /home/oslab/a.txt
5539 : /home/oslab/testdir
5545 : /home/oslab/testdir2
% ./ssu_sync
20220000〉 remove 5539
monitoring ended (/home/oslab/testdir) : 5539
20220000〉 exit
% cat ~/backup/monitor_list.log
5230 : /home/oslab/a.txt
5545 : /home/oslab/testdir2
% cat ~/backup/5539.log
cat: ~/backup/5539.log: No such file or directory
% ls /home/oslab/backup/5539
ls: cannot access '/home/oslab/backup/5539': No such file or directory
```

4) 예외 처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)
- 첫 번째 인자 〈DAEMON_PID〉가 "monitor_list.log"에 존재하지 않는 경우 에러처리 후 프롬프트 재출력

### 4. 내장명령어 3. list

1) Usage : list [DAEMON_PID]
- 현재 모니터링 중인 데몬 프로세스 혹은 데몬 프로세스가 관리중인 경로에 대해 트리 형태로 출력함

2) 인자 설명
- 첫 번째 인자 〈DAEMON_PID〉는 현재 모니터링을 진행 중인 데몬 프로세스의 pid 중 하나이며 생략 가능함

3) 실행결과
- 첫 번째 인자 생략 시 현재 실행중인 데몬 프로세스들의 목록을 출력함
- 첫 번째 인자로 데몬 프로세스의 pid를 입력 시 해당 데몬 프로세스가 관리 중인 경로에 대해 트리 형태로 출력함
- 데몬 프로세스가 관리 중인 경로에 대해 각 정규 파일들은 아래 구조와 같이 해당 데몬 프로세스 로그를 기준으로 변경 상태와 수행 시간을 트리 형태로 출력함

예제 4. list 내장명령어 실행결과(현재 작업 디렉토리(pwd)가 "/home/oslab"일 경우)

```
% cat /home/oslab/backup/5539.log
[2024-05-03 10:00:00][create][/home/oslab/testdir/a/a.txt]
[2024-05-03 10:00:00][create][/home/oslab/testdir/a.txt]
[2024-05-03 10:00:00][create][/home/oslab/testdir/b.txt]
[2024-05-03 10:00:00][create][/home/oslab/testdir/c.txt]
[2024-05-03 10:00:30][modify][/home/oslab/testdir/a.txt]
[2024-05-03 10:00:43][remove][/home/oslab/testdir/b.txt]
[2024-05-03 10:01:07][create][/home/oslab/testdir/a/b/a.txt]
[2024-05-03 10:01:24][create][/home/oslab/testdir/d.txt]
% ./ssu_sync
20220000〉 list
5230 : /home/oslab/a.txt
5539 : /home/oslab/testdir
5545 : /home/oslab/testdir2
20220000〉 list 5539
/home/oslab/testdir
├ a/
|  ├ a.txt
|  |  └ [create] [2024-05-03 10:00:00]
|  └ b/
|     └ a.txt~~_20240503100107~~
|        └ [create] [2024-05-03 10:01:07]
```

```
├ a.txt_20240503100000
|  ├ [create] [2024-05-03 10:00:00]
|  └ [modify] [2024-05-03 10:00:30]
├ b.txt_20240503100000
|  ├ [create] [2024-05-03 10:00:00]
|  └ [remove] [2024-05-03 10:00:43]
├ c.txt_20240503100000
|  └ [create] [2024-05-03 10:00:00]
└ d.txt_20240503100124
   └ [create] [2024-05-03 10:01:24]
```

4) 예외 처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)
- 첫 번째 인자 〈DAEMON_PID〉가 “monitor_list.log”에 존재하지 않는 경우 에러처리 후 프롬프트 재출력
- 현재 모니터링중인 데몬 프로세스가 존재하지 않을 시 에러처리 후 프롬프트 재출력

**5. 내장명령어 4. help**

1) Usage : help [COMMAND]
- 프로그램 내장명령어에 대한 설명(Usage) 출력

2) 인자 설명
- 첫 번째 인자 [COMMAND]에 대해 해당 내장명령어에 대한 설명(Usage)를 출력. 첫 번째 인자는 생략 가능하며, 생략 시 모든 내장명령어에 대한 설명(Usage) 출력

3) 실행결과

예제 5. help 내장명령어 실행

```
% ./ssu_sync
20220000〉 help
Usage:
  〉 add 〈PATH〉 [OPTION]... : add new daemon process of 〈PATH〉 if 〈PATH〉 is file
    -d : add new daemon process of 〈PATH〉 if 〈PATH〉 is directory
    -r : add new daemon process of 〈PATH〉 recursive if 〈PATH〉 is directory
    -t 〈TIME〉 : set daemon process time to 〈TIME〉 sec (default : 1sec)
  〉 remove 〈DAEMON_PID〉 : delete daemon process with 〈DAEMON_PID〉
  〉 list [DAEMON_PID] : show daemon process list or dir tree
  〉 help [COMMAND] : show commands for program
  〉 exit : exit program

20220000〉 help add
Usage: add 〈PATH〉 [OPTION]... : add new daemon process of 〈PATH〉 if 〈PATH〉 is file
  -d : add new daemon process of 〈PATH〉 if 〈PATH〉 is directory
  -r : add new daemon process of 〈PATH〉 recursive if 〈PATH〉 is directory
  -t 〈TIME〉 : set daemon process time to 〈TIME〉 sec (default : 1sec)

20220000〉
```

4) 예외 처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)
- 잘못된 내장명령어 입력 시 에러 처리 후 프롬프트 재출력

**6. 내장명령어 5. exit**

1) Usage : exit
- 현재 실행중인 ssu_sync 프로그램 종료

2) 실행결과
- 프로그램 종료

예제 6. exit 내장명령어 실행

```
% ./ssu_sync
20220000〉 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 : 오류가 발생, 상세한 오류 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 : 오류가 발생, 상세한 오류 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 함수(라이브러리, 시스템콜)

```
#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[]);
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. MD5

✓ MD5 해시값을 구하기 위해 MD5(openssl/md5.h)를 사용

- Linux, Ubuntu :"sudo apt-get install libssl-dev"로 라이브러리 설치 필요
- Linux, Fedora :"sudo dnf-get install libssl-devel"로 라이브러리 설치 필요
- MacOS : homebrew (https://brew.sh/ 참고) 설치 -〉 % brew install openssl
- MD5 관련 함수를 사용하기 위해 컴파일 시 "-lcrypto" 옵션 필요
- md5() 사용법은 https://www.openssl.org/docs/man1.1.1/man3/MD5.html 및 https://github.com/Chronic-Dev/openssl/blob/master/crypto/md5/md5.c 참고

○ make와 Makefile

- make : 프로젝트 관리 유틸리티
  - ✓ 파일에 대한 반복 명령어를 자동화하고 수정된 소스 파일만 체크하여 재컴파일 후 종속된 부분만 재링크함
  - ✓ Makefile(규칙을 기술한 파일)에 기술된 대로 컴파일 명령 또는 쉘 명령을 순차적으로 실행함
- Makefile의 구성
  - ✓ Macro(매크로) : 자주 사용되는 문자열 또는 변수 정의 (컴파일러, 링크 옵션, 플래그 등)
  - ✓ Target(타겟) : 생성할 파일
  - ✓ Dependency(종속 항목) : 타겟을 만들기 위해 필요한 파일의 목록
  - ✓ Command(명령) : 타겟을 만들기 위해 필요한 명령(shell)

```
Macro

Target : Dependency1 Dependency2 ...
〈-Tab-〉Command 1
〈-Tab-〉Command 2
〈-Tab-〉...
```

- Makefile의 동작 순서
  - ✓ make 사용 시 타겟을 지정하지 않으면 제일 처음의 타겟을 수행
  - ✓ 타겟과 종속 항목들은 관습적으로 파일명을 명시
  - ✓ 명령 항목들이 충족되었을 때 타겟을 생성하기 위해 명령 (command 라인의 맨 위부터 순차적으로 수행)
  - ✓ 종속 항목의 마지막 수정 시간(st_mtime)을 비교 후 수행
- Makefile 작성 시 참고사항
  - ✓ 명령의 시작은 반드시 Tab으로 시작해야함
  - ✓ 한 줄 주석은 #, 여러 줄 주석은 자동 매크로 : 현재 타겟의 이름이나 종속 파일을 표현하는 매크로

| 매크로 | 설명 |
|---|---|
| $? | 타겟보다 최근에 변경된 종속 항목 리스트 (확장자 규칙에서 사용 불가능) |
| $^ | 현재 타겟의 종속 항목 (확장자 규칙에서 사용 불가능) |
| $< | 타겟보다 최근에 변경된 종속 항목 리스트 (확장자 규칙에서만 사용 가능) |
| $* | 타겟보다 최근에 변경된 종속 항목 리스트 (확장자 규칙에서만 사용 가능) |
| $@ | 현재 타겟의 이름 |

- Makefile 작성 예시

(예 1). Makefile

```
test : test.o add.o sub.o
    gcc test.o add.o sub.o -o test

test.o: test.c
    gcc -c test.c

add.o: add.c
    gcc -c add.c

sub.o: sub.c
    gcc -c sub.c

clean :
    rm test.o
    rm add.o
    rm sub.o
    rm test
```

(예 2). 매크로를 사용한 경우

```
OBJECTS = test.o add.o sub.o
TARGET = test
CC = gcc

$(TARGET) : $(OBJECTS)
    $(CC) -o $(TARGET) $(OBJECTS)

test.o: test.c
    $(CC) -c test.c

add.o: add.c
    $(CC) -c add.c

sub.o: sub.c
    $(CC) -c sub.c
```

(예 3). 자동 매크로를 사용한 경우

```
OBJECTS = test.o add.o sub.o
TARGET = test
CC = gcc

$(TARGET) : $(OBJECTS)
    $(CC) -o $@ $^

test.o: test.c
    $(CC) -c $^

add.o: add.c
    $(CC) -c $^

sub.o: sub.c
    $(CC) -c $^
```

- (예 4). Makefile 수행 예시

```
oslab@a-VirtualBox:~$ make
gcc -c test.c
gcc -c add.c
gcc -c sub.c
gcc test.o add.o sub.o -o test
oslab@a-VirtualBox:~$
```

```
oslab@a-VirtualBox:~$ make clean
rm test.o
rm add.o
rm sub.o
rm test
oslab@a-VirtualBox:~$
```

○ 보고서 제출 시 유의 사항

- 보고서 제출 마감은 제출일 11:59PM까지 (구글 서버시간)
- 지연 제출 시 감점 : 1일 지연 시 30% 감점, 2일 이후 미제출 처리
- 압축 오류, 파일 누락 관련 감점 syllabus 참고

- 필수구현 : 1, 2('-r', '-t' 옵션은 필수기능 아님), 3, 4, 5, 6(예외처리는 필수구현 아님. 단, 별도 감점 있음)
- 배점(100점 만점. 실행 여부 배점 80점으로 최종 환산하며, 보고서 15점과 소스코드 주석 5점은 별도, 강의계획서 확인)

1. ssu_sync - 10점
2. 내장명령어 1. add - 60점
   - ✓ '-d' 옵션 (15점)
   - ✓ '-r' 옵션 (15점)
   - ✓ '-t' 옵션 (15점)
3. 내장명령어 2. remove - 5점
4. 내장명령어 3. list - 15점
5. 내장명령어 4. help - 3점
6. 내장명령어 5. exit - 2점

- makefile 작성(매크로 사용하지 않아도 됨) - 5점