설계과제 1 개요 : SSU-Backup

○ 개요
- 리눅스 시스템 상에서 사용자가 백업을 원하는 파일이나 디렉토리를 옵션에 따라 백업하고 백업된 파일을 다시 복원 및 삭제하고, 백업 상태를 관리하는 프로그램

○ 목표
- 새로운 명령어를 시스템 함수를 사용하여 구현함으로써 쉘의 원리를 이해하고, 유닉스/리눅스 시스템에서 제공하는 여러 시스템 자료구조와 시스템 콜 및 라이브러리를 이용하여 프로그램을 작성함으로써 표준 입출력 및 파일 입출력에 대해 적응하고 이를 응용하여 디렉토리 구조를 링크드 리스트로 구현하며 시스템 프로그래밍 설계 및 응용 능력을 향상시킴

○ 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
- 3월 31일(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_backup 프로그램 기본 사항
- 프로그램 실행 시 /home 디렉토리 내에 /backup 디렉토리가 없다면 생성
- 프로그램 실행 시 백업 디렉토리(/home/backup) 내에 로그 파일(ssubak.log)이 없다면 생성
- 백업 디렉토리 접근을 위해 root권한으로 프로그램 실행해야 함

- 프로그램 실행 시 백업 디렉토리 내에 모든 백업 파일에 대해 로그 파일 내용과 대조하여 파일들에 대한 백업 상태를 링크드 리스트로 관리
  ✓ 리눅스 상에서 파일 경로의 최대 크기는 4,096 바이트이며, 파일 이름의 최대 크기는 255 바이트임
- 프로그램 실행 시 내장명령어(backup, remove, recover, list, help)에 따라 해당 기능 실행
- ssu_backup 프로그램을 통해 백업 가능한 경로들은 사용자 홈 디렉토리(/home/사용자아이디)내 경로여야 함
- 백업 파일과 원본 파일에 대한 비교는 md5를 이용하여 해시값을 통해 비교
- 프로그램 전체에서 system() 절대 사용 불가. 사용 시 0점 처리
- getopt() 라이브러리를 사용하여 옵션 처리 권장
- ssu_backup은 foreground로만 수행되는 것으로 가정함(& background 수행은 안되는 것으로 함)
- 본문에서의 예제 및 그림은 대략적인 실행결과를 나타냄. 이 외 예외처리 및 다양한 상황에 대해서는 구글 클래스룸에 올라온 '예제 모음.pdf' 참고. 에러 처리 형식에 대해서는 무슨 에러인지에 대해 출력하고 형식은 자유

○ 설계 및 구현

1. ssu_backup

1) Usage : ssu_backup 〈backup|remove|recover|list|help〉 〈...〉
- ssu_backup 실행 시 인자로 내장명령어 및 내장명령어에 따른 인자, 옵션 입력
- ssu_backup 프로그램의 내장명령어는 백업(backup), 백업파일 제거(remove), 백업 복구(recover), 백업 목록(list), 내장명령어 설명(help)로 이루어져 있음
- 각 내장명령어에 따라 필요한 인자 및 옵션 또한 프로그램 인자를 통해 입력받음

2) 인자 설명
- 첫 번째 인자는 프로그램의 내장명령어를 입력받음
- 두 번째 인자부터는 각 내장명령어에 따라 경로 및 옵션을 입력받음

3) 실행결과
- ssu_backup의 실행경과는 각 명령어의 실행 결과를 출력함
- ssu_backup 프로그램을 처음 실행 시 혹은 /home 디렉토리 아래 backup 디렉토리가 존재하지 않을 시 backup 디렉토리를 생성함. 더하여 백업 디렉토리 아래 ssubak.log 파일이 없다면 생성함. 이미 존재할 경우에는 다시 생성하지 않음

예제 1. ssu_backup 실행결과(help를 통해 명령어 목록을 출력한 모습)

```
% sudo ./ssu_backup help
Usage:
  〉backup 〈PATH〉 [OPTION]... : backup file if 〈PATH〉 is file
    -d : backup files in directory if 〈PATH〉 is directory
    -r : backup files in directory recursive if 〈PATH〉 is directory
    -y : backup file although already backuped
  〉remove 〈PATH〉 [OPTION]... : remove backuped file if 〈PATH〉 is file
    -d : remove backuped files in directory if 〈PATH〉 is directory
    -r : remove backuped files in directory recursive if 〈PATH〉 is directory
    -a : remove all backuped files
  〉recover 〈PATH〉 [OPTION]... : recover backuped file if 〈PATH〉 is file
    -d : recover backuped files in directory if 〈PATH〉 is directory
    -r : recover backuped files in directory recursive if 〈PATH〉 is directory
    -l : recover latest backuped file
    -n 〈NEW_PATH〉 : recover backuped file with new path
  〉list [PATH] : show backup list by directory structure
    〉〉 rm 〈INDEX〉 [OPTION]... : remove backuped files of [INDEX] with [OPTION]
    〉〉 rc 〈INDEX〉 [OPTION]... : recover backuped files of [INDEX] with [OPTION]
    〉〉 vi(m) 〈INDEX〉 : edit original file of [INDEX]
    〉〉 exit : exit program
  〉help [COMMAND] : show commands for program
```

4) 예외처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)
- 첫 번째 인자를 입력하지 않을 경우(명령어를 입력하지 않을 경우) 에러 처리 후 프로그램 비정상 종료(2점)
- 첫 번째 인자로 잘못된 명령어를 입력할 경우 에러 처리 후 프로그램 비정상 종료(2점)

2. 내장명령어 1. backup

1) Usage : backup 〈PATH〉 [OPTION]...
- 백업할 경로(PATH)를 입력받아 해당 파일 또는 디렉토리를 백업함

2) 인자 설명
- 첫 번째 인자 〈PATH〉은 백업할 파일이나 디렉토리의 상대경로와 절대경로 모두 입력 가능해야 함
- 두 번째 인자 [OPTION]은 '-d', '-r', '-y'가 있으며 동시 사용 가능하고, 생략 가능함('-d', '-r', '-y' 옵션은 아래 설명 확인)

3) 실행결과

- 첫 번째 인자 〈PATH〉를 백업 디렉토리에 백업함. 백업 진행 시 백업 시간(YYMMDDHHMMSS)을 이름으로 하는 버전 디렉토리를 백업 디렉토리 내에 만들고 버전 디렉토리 내에 백업한 파일, 디렉토리를 복사하여 붙여넣음. 명령어 실행 후 백업한 파일이 없다면 버전 디렉토리를 만들지 않음
- 첫 번째 인자 〈PATH〉의 최하위 디렉토리를 기준으로 버전 디렉토리와 연동하여 백업을 진행함 (ex. /home/oslab/a/b.txt를 백업할 시 새로운 버전 디렉토리(/home/backup/백업시간)은 /home/oslab/a/와 연동)
- 백업 성공 시 원본 경로와 백업 경로, 백업 시간을 ("원본 경로" backuped to "백업 경로")의 형태로 출력하고 백업 로그(/home/backup/ssubak.log)에 (백업시간 : "원본 경로" backuped to "백업 경로")의 형태로 append하여 저장
- 이미 내용이 같은 백업파일이 존재할 경우 ("원본 경로" already backuped to "백업 경로")의 형태로 출력하고 따로 로그파일에는 해당 내용을 작성하지 않음. 내용이 같은 백업파일이 여러개일 경우 가장 최근에 백업한 파일의 경로를 출력

예제 2-1-1. backup 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup backup a.txt
"/home/oslab/a.txt" backuped to "/home/backup/240225163125/a.txt"

% sudo ./ssu_backup backup ./a/b/c.txt
"/home/oslab/a/b/c.txt" backuped to "/home/backup/240225163137/c.txt"

//절대 경로와 상대경로 모두 입력 가능 a.txt == /home/oslab/a.txt
% sudo ./ssu_backup backup /home/oslab/a.txt
"/home/oslab/a.txt" already backuped to "/home/backup/240225163125/a.txt"

% echo "hello world" 〉 a.txt

% sudo ./ssu_backup backup ./a.txt
"/home/oslab/a.txt" backuped to "/home/backup/240225163207/a.txt"
```

예제 2-1-2. 백업 로그(ssubak.log) 파일 내용

```
...
240225163125 : "/home/oslab/a.txt" backuped to "/home/backup/240225163125/a.txt"
240225163137 : "/home/oslab/a/b/c.txt" backuped to "/home/backup/240225163137/c.txt"
240225163207 : "/home/oslab/a.txt" backuped to "/home/backup/240225163207/a.txt"
```

그림 2-1-1. /home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
|  ├ a.txt
|  ├ b/
|  |  └ c.txt
|  └ c.txt
├ a.txt
└ b/
```

그림 2-1-2. 백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
|  └ a.txt
├ 240225163137/
|  └ c.txt
├ 240225163207/
|  └ a.txt
└ ssubak.log
```

- '-d' 옵션 입력시에는 〈PATH〉가 디렉토리일 시 해당 경로 아래 있는 모든 파일들에 대해 백업을 진행함.

예제 2-2-1. -d 옵션과 함께 backup 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup backup ./a -d
"/home/oslab/a/a.txt" backuped to "/home/backup/240225163125/a.txt"
"/home/oslab/a/c.txt" backuped to "/home/backup/240225163125/c.txt"
```

그림 2-2-1. /home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
└ a/
   ├ a.txt
   ├ b/
   |  └ c.txt
   └ c.txt
```

그림 2-2-2. 백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
|  ├ a.txt
|  └ c.txt
└ ssubak.log
```

- '-r' 옵션 입력시에는 〈PATH〉가 디렉토리일 시 해당 경로 아래 있는 모든 파일들에 대해 백업을 진행함. 이 때, 서브 디렉토리 내에 모든 파일들에 대해서도 재귀적으로 탐색하여 백업을 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 백업을 진행함

예제 2-3-1. -r 옵션과 함께 backup 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup backup ./a -r
"/home/oslab/a/a.txt" backuped to "/home/backup/240225163125/a.txt"
"/home/oslab/a/c.txt" backuped to "/home/backup/240225163125/c.txt"
"/home/oslab/a/b/c.txt" backuped to "/home/backup/240225163125/b/c.txt"
```

그림 2-3-1. /home/oslab/ 디렉토리 트리 구조	그림 2-3-2. 백업 디렉토리(/home/backup/) 트리 구조

/home/oslab/
└ a/
  ├ a.txt
  ├ b/
  │ └ c.txt
  └ c.txt
	/home/backup/
	├ 240225163125/
	│ ├ a.txt
	│ ├ b/
	│ │ └ c.txt
	│ └ c.txt
	└ ssubak.log

- '-y' 옵션 입력시에는 〈PATH〉에 대해 경로와 내용이 같은 백업파일이 존재해도 백업을 진행함

예제 2-4-1. -y 옵션과 함께 backup 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup backup ./a.txt
"/home/oslab/a.txt" backuped to "/home/backup/240225163125/a.txt"

% sudo ./ssu_backup backup ./a.txt
"/home/oslab/a.txt" already backuped to "/home/backup/240225163125/a.txt"

% sudo ./ssu_backup backup ./a.txt -y
"/home/oslab/a.txt" backuped to "/home/backup/240225163148/a.txt"
```

그림 2-4-1. /home/oslab/ 디렉토리 트리 구조	그림 2-4-2. 백업 디렉토리(/home/backup/) 트리 구조
/home/oslab/ ├ a/ │ ├ a.txt │ └ c.txt ├ a.txt └ b/	/home/backup/ ├ 240225163125/ │ └ a.txt ├ 240225163148/ │ └ a.txt └ ssubak.log

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

- 경로를 입력하지 않을 경우, backup 명령어에 대한 Usage 출력 후 프로그램 비정상 종료(1점)
- 인자로 입력받은 경로가 길이 제한(4,096 Byte)를 넘거나 경로가 올바르지 않은 경우, 에러 처리 후 프로그램 비정상 종료(1점)
- 인자로 입력받은 경로가 일반 파일이나 디렉토리가 아니거나 해당 경로에 대해 접근권한이 없는 경우, 에러 처리 후 프로그램 비정상 종료(2점)
- 인자로 입력받은 경로가 사용자의 홈 디렉토리를 벗어날 경우, 에러 처리 후 프로그램 비정상 종료(2점)
- 인자로 입력받은 옵션이 올바르지 않을 경우, backup 명령어에 대한 Usage 출력 후 프로그램 비정상 종료(1점)
- 인자로 입력받은 옵션 중 -d, -r을 사용하지 않았는데, 인자로 입력받은 경로가 디렉토리일 경우, 에러 처리 후 프로그램 비정상 종료(2점)
- 인자로 입력받은 옵션 중 -d, -r을 사용하였는데, 인자로 입력받은 경로가 파일일 경우, 에러 처리 후 프로그램 비정상 종료(2점)
- 인자로 입력받은 경로에 대해 기존 경로가 같고, 내용이 같은 파일이 이미 백업 디렉토리에 존재할 경우, 백업을 하지 않고, ("원본 경로" already backuped to "백업 경로")를 표준 출력(5점)

3. 내장명령어 2. remove

1) Usage : remove 〈PATH〉 [OPTION]...
- 원본 경로(PATH)를 입력받아 해당 경로와 일치하는 백업 파일 또는 디렉토리를 삭제함

2) 인자 설명
- 첫 번째 인자 〈PATH〉은 파일이나 디렉토리의 상대경로와 절대경로 모두 입력 가능해야 함
- 두 번째 인자 [OPTION]은 '-d', '-r', '-a'가 있으며 동시 사용 가능하고, 생략 가능함('-d', '-r', '-a' 옵션은 아래 설명 확인)

3) 실행결과
- 첫 번째 인자 〈PATH〉에 대해 백업 디렉토리에 해당 경로에 대한 백업 파일을 삭제함. 〈PATH〉에 대한 백업 파일이 여러개일 경우 백업 파일에 대한 정보들을 번호를 매겨 출력. 목록 맨 위에는 (backup files of "원본 경로")를 출력하고 다음 줄부터 번호를 매겨 백업 파일에 대한 백업 시간, 파일 크기(천 단위로 쉼표 출력)를 출력함. 이 때 목록에 대한 출력 순서는 백업 시간에 대해 오름차순으로 정렬.
- 사용자 입력에 대해 번호를 입력하면 해당 번호에 맞는 백업 파일에 대해 삭제를 진행. 사용자 입력에 대해 0번 입력 시 백업 파일에 대한 삭제를 진행하지 않고 탈출
- 삭제 성공 시 ("백업 경로" removed by "원본 경로")의 형태로 출력하고 백업 로그(/home/backup/ssubak.log)에 (삭제 시간 : "백업 경로" removed by "원본 경로")의 형태로 append하여 저장
- 명령어 실행 후 해당 버전 디렉토리 안에 아무 파일이 없다면 해당 버전 디렉토리 삭제

예제 3-1-1. remove 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup remove ./a.txt
backup files of /home/oslab/a.txt
0. exit
1. 240225163125          14bytes
2. 240225163207          11bytes
>> 0

% sudo ./ssu_backup remove ./a.txt
backup files of /home/oslab/a.txt
0. exit
1. 240225163125          14bytes
2. 240225163207          11bytes
>> 1
"/home/backup/240225163125/a.txt" removed by "/home/oslab/a.txt"

% sudo ./ssu_backup remove ./a.txt
"/home/backup/240225163207/a.txt" removed by "/home/oslab/a.txt"

% sudo ./ssu_backup remove ./a/c.txt
"/home/backup/240225163137/c.txt" removed by "/home/oslab/a/c.txt"
```

예제 3-1-2. 백업 로그(ssubak.log) 파일 내용

```
...
240225163302 : "/home/backup/240225163125/a.txt" removed by "/home/oslab/a.txt"
240225163314 : "/home/backup/240225163207/a.txt" removed by "/home/oslab/a.txt"
240225163343 : "/home/backup/240225163137/c.txt" removed by "/home/oslab/a/c.txt"
```

그림 3-1-1. /home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
|  ├ a.txt
|  └ b/
|     └ c.txt
├ a.txt
└ b/
```

그림 3-1-2. (전)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
|  └ a.txt
├ 240225163137/
|  └ c.txt
├ 240225163207/
|  └ a.txt
└ ssubak.log
```

그림 3-1-3. (후)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
└ ssubak.log
```

- '-d' 옵션 입력 시에는 〈PATH〉가 디렉토리일 시 해당 경로 아래 있는 모든 파일들의 백업 파일에 대해 삭제를 진행함.
- 백업 파일이 2개 이상인 파일들에 대해 모두 목록을 출력하여 사용자 입력을 기다림

예제 3-2-1. remove 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup remove /home/oslab/a -d
"/home/backup/240225163125/a.txt" removed by "/home/oslab/a/a.txt"
backup files of /home/oslab/a/c.txt
0. exit
1. 240225163125          14bytes
2. 240225163207          11bytes
>> 2
"/home/backup/240225163207/c.txt" removed by "/home/oslab/a/c.txt"
```

그림 3-2-1. /home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
|  ├ a.txt
|  └ b/
|     └ c.txt
├ a.txt
└ b/
```

그림 3-2-2. (전)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
|  ├ a.txt
|  ├ b/
|  |  └ c.txt
|  └ c.txt
├ 240225163207/
|  └ c.txt
└ ssubak.log
```

그림 3-2-3. (후)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
|  ├ b/
|  |  └ c.txt
|  └ c.txt
└ ssubak.log
```

- '-r' 옵션 입력 시에는 〈PATH〉가 디렉토리일 시 해당 경로 아래 있는 모든 파일들의 백업 파일에 대해 삭제를 진행함. 이 때, 서브 디렉토리 내에 모든 파일들에 대해서도 재귀적으로 탐색하여 백업 파일에 대한 삭제를 진행함. 진행 순서는 파일을 우선으로 하고 그 다음 서브디렉토리를 재귀적으로 탐색함(BFS)
- 백업 파일이 2개 이상인 파일들에 대해 모두 목록을 출력하여 사용자 입력을 기다림

- '-r' 옵션과 '-d' 옵션을 동시 입력 시에는 '-r' 옵션을 적용시켜 재귀적으로 백업 파일에 대한 삭제를 진행함

예제 3-3-1. remove 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup remove /home/oslab/a -r
"/home/backup/240225163125/a.txt" removed by "/home/oslab/a/a.txt"
backup files of /home/oslab/a/c.txt
0. exit
1. 240225163125          14bytes
2. 240225163207          11bytes
>> 2
"/home/backup/240225163207/c.txt" removed by "/home/oslab/a/c.txt"
"/home/backup/240225163125/b/c.txt" removed by "/home/oslab/a/b/c.txt"
```

그림 3-3-1. /home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
|  ├ a.txt
|  └ b/
|     └ c.txt
├ a.txt
└ b/
```

그림 3-3-2. (전)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
|  ├ a.txt
|  ├ b/
|  |  └ c.txt
|  └ c.txt
├ 240225163207/
|  └ c.txt
└ ssubak.log
```

그림 3-2-3. (후)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
|  └ c.txt
└ ssubak.log
```

- '-a' 옵션 입력 시에는 백업본이 2개 이상인 파일들에 대해 목록을 출력하지 않고 모든 백업 파일을 삭제함.

예제 3-4-1. remove 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup remove /home/oslab/a -r -a
"/home/backup/240225163125/a.txt" removed by "/home/oslab/a/a.txt"
"/home/backup/240225163125/c.txt" removed by "/home/oslab/a/c.txt"
"/home/backup/240225163207/c.txt" removed by "/home/oslab/a/c.txt"
"/home/backup/240225163125/b/c.txt" removed by "/home/oslab/a/b/c.txt"
```

그림 3-4-1. /home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
|  ├ a.txt
|  └ b/
|     └ c.txt
├ a.txt
└ b/
```

그림 3-4-2. (전)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
|  ├ a.txt
|  ├ b/
|  |  └ c.txt
|  └ c.txt
├ 240225163207/
|  └ c.txt
└ ssubak.log
```

그림 3-2-3. (후)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
└ ssubak.log
```

4) 예외 처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)
- 경로를 입력하지 않을 경우, remove 명령어에 대한 Usage 출력 후 프로그램 비정상 종료(1점)
- 인자로 입력받은 경로가 길이 제한(4,096 Byte)를 넘거나 경로가 올바르지 않은 경우, 에러 처리 후 프로그램 비정상 종료(1점)
- 인자로 입력받은 경로와 일치하는 백업 파일이나 디렉토리가 없는 경우, 에러 처리 후 프로그램 비정상 종료(2점)
- 인자로 입력받은 경로가 사용자의 홈 디렉토리를 벗어날 경우, 에러 처리 후 프로그램 비정상 종료(2점)
- 인자로 입력받은 옵션이 올바르지 않을 경우, backup 명령어에 대한 Usage 출력 후 프로그램 비정상 종료(1점)
- 인자로 입력받은 옵션 중 -d, -r을 사용하지 않았는데, 인자로 입력받은 경로가 존재하고 디렉토리일 경우, 에러 처리 후 프로그램 비정상 종료(2점)
- 인자로 입력받은 옵션 중 -d, -r을 사용하였는데, 인자로 입력받은 경로가 존재하고 파일일 경우, 에러 처리 후 프로그램 비정상 종료(2점)

4. 내장명령어 3. recover

1) Usage : recover 〈PATH〉 [OPTION]...
- 원본 경로(PATH)를 입력받아 해당 경로와 일치하는 백업 파일 또는 디렉토리를 복구함

2) 인자 설명
- 첫 번째 인자 〈PATH〉은 파일이나 디렉토리의 상대경로와 절대경로 모두 입력 가능해야 함
- 두 번째 인자 [OPTION]은 '-d', '-r', '-l', '-n'가 있으며 동시 사용 가능하고, 생략 가능함('-d', '-r', '-l', '-n' 옵션은 아래 설명 확인)

3) 실행결과

- 첫 번째 인자 〈PATH〉에 대해 백업 디렉토리에 해당 경로에 대한 백업 파일을 원본 경로에 복구하고(원본 경로에 파일이 있을 경우 덮어씌움) 백업 파일을 삭제함. 〈PATH〉에 대한 백업 파일이 여러개일 경우 백업 파일에 대한 정보들을 번호를 매겨 출력. 목록 맨 위에는 (backup files of "원본 경로")를 출력하고 다음 줄부터 번호를 매겨 백업 파일에 대한 백업 시간, 파일 크기(천 단위로 쉼표 출력)를 출력함. 이 때 목록에 대한 출력 순서는 백업 시간에 대해 오름차순으로 정렬
- 사용자 입력에 대해 번호를 입력하면 해당 번호에 맞는 백업 파일에 대해 복구를 진행. 사용자 입력에 대해 0번 입력 시 백업 파일에 대한 복구를 진행하지 않고 탈출
- 복구 성공 시 ("백업 경로" recovered to "원본 경로")의 형태로 출력하고 백업 로그 (/home/backup/ssubak.log)에 (복구시간 : "백업 경로" recovered to "원본 경로")의 형태로 append하여 저장
- 명령어 실행 후 해당 버전 디렉토리 안에 아무 파일이 없다면 해당 버전 디렉토리 삭제

예제 4-1-1. recover 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup backup ./a.txt
"/home/oslab/a.txt" backuped to "/home/backup/240225163207/a.txt"

% sudo ./ssu_backup recover ./a.txt
backup files of /home/oslab/a.txt
0. exit
1. 240225163125          14bytes
2. 240225163207          11bytes
〉〉 0

% sudo ./ssu_backup recover ./a.txt
backup files of /home/oslab/a.txt
0. exit
1. 240225163125          14bytes
2. 240225163207          11bytes
〉〉 2
"/home/backup/240225163207/a.txt" not changed with "/home/oslab/a.txt"

% sudo ./ssu_backup recover ./a.txt
backup files of /home/oslab/a.txt
0. exit
1. 240225163125          14bytes
2. 240225163207          11bytes
〉〉 1
"/home/backup/240225163125/a.txt" recovered to "/home/oslab/a.txt"

% sudo ./ssu_backup recover ./a.txt
"/home/backup/240225163207/a.txt" recovered to "/home/oslab/a.txt"

% sudo ./ssu_backup recover ./a/c.txt
"/home/backup/240225163137/c.txt" recovered to "/home/oslab/a/c.txt"
```

예제 4-1-2. 백업 로그(ssubak.log) 파일 내용

```
...
240225163207 : "/home/oslab/a.txt" backuped to "/home/backup/240225163207/a.txt"
240225163302 : "/home/backup/240225163207/a.txt" recovered to "/home/oslab/a.txt"
240225163314 : "/home/backup/240225163125/a.txt" recovered to "/home/oslab/a.txt"
240225163343 : "/home/backup/240225163137/c.txt" recovered to "/home/oslab/a/c.txt"
```

그림 4-1-1. (전)/home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
|  ├ a.txt
|  └ b/
|     └ c.txt
├ a.txt
└ b/
```

그림 4-1-2. (전)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
|  └ a.txt
├ 240225163137/
|  └ c.txt
├ 240225163207/
|  └ a.txt
└ ssubak.log
```

그림 4-1-3. (후)/home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
|  ├ a.txt
|  ├ b/
|  |  └ c.txt
|  └ c.txt
├ a.txt
└ b/
```

그림 4-1-4. (후)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
└ ssubak.log
```

- '-d' 옵션 입력 시에는 〈PATH〉가 디렉토리일 시 해당 경로 아래 있는 모든 파일들의 백업 파일에 대해 복구를 진행함.
- 백업 파일이 2개 이상인 파일들에 대해 모두 목록을 출력하여 사용자 입력을 기다림

예제 4-2-1. recover 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup recover /home/oslab/a -d
"/home/backup/240225163125/a.txt" recovered to "/home/oslab/a/a.txt"
backup files of /home/oslab/a/c.txt
0. exit
1. 240225163125          14bytes
2. 240225163207          11bytes
>> 2
"/home/backup/240225163207/c.txt" recovered to "/home/oslab/a/c.txt"
```

그림 4-2-1. (전)/home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
│  ├ a.txt
│  └ b/
│     └ c.txt
├ a.txt
└ b/
```

그림 4-2-3. (후)/home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
│  ├ a.txt
│  ├ b/
│  │  └ c.txt
│  └ c.txt
├ a.txt
└ b/
```

그림 4-2-2. (전)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
│  ├ a.txt
│  ├ b/
│  │  └ c.txt
│  └ c.txt
├ 240225163207/
│  └ c.txt
└ ssubak.log
```

그림 4-2-4. (후)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
│  ├ b/
│  │  └ c.txt
│  └ c.txt
└ ssubak.log
```

- '-r' 옵션 입력 시에는 〈PATH〉가 디렉토리일 시 해당 경로 아래 있는 모든 파일들의 백업 파일에 대해 복구를 진행함. 이 때, 서브디렉토리 내에 모든 파일들에 대해서도 재귀적으로 탐색하여 백업을 진행함. 진행 순서는 파일을 우선으로 하고 그 다음 서브디렉토리를 재귀적으로 탐색함(BFS)
- 백업 파일이 2개 이상인 파일들에 대해 모두 목록을 출력하여 사용자 입력을 기다림
- '-r' 옵션과 '-d' 옵션을 동시 입력 시에는 '-r' 옵션을 적용시켜 재귀적으로 백업 파일에 대한 복구를 진행함

예제 4-3-1. recover 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup recover /home/oslab/a -r
"/home/backup/240225163125/a.txt" recovered to "/home/oslab/a/a.txt"
backup files of /home/oslab/a/c.txt
0. exit
1. 240225163125          14bytes
2. 240225163207          11bytes
>> 2
"/home/backup/240225163207/c.txt" recovered to "/home/oslab/a/c.txt"
"/home/backup/240225163125/b/c.txt" recovered to "/home/oslab/a/b/c.txt"
```

그림 4-3-1. (전)/home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
│  ├ a.txt
│  └ b/
│     └ c.txt
├ a.txt
└ b/
```

그림 4-3-3. (후)/home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
│  ├ a.txt
│  ├ b/
│  │  └ c.txt
│  └ c.txt
├ a.txt
└ b/
```

그림 4-3-2. (전)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
│  ├ a.txt
│  ├ b/
│  │  └ c.txt
│  └ c.txt
├ 240225163207/
│  └ c.txt
└ ssubak.log
```

그림 4-3-4. (후)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
│  └ c.txt
└ ssubak.log
```

- '-l' 옵션 입력 시에는 백업 파일이 2개 이상인 파일들에 대해 리스트를 출력하지 않고 가장 최근에 백업한 백업 파일을 복구함.

예제 4-4-1. recover 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup recover ./a.txt
backup files of /home/oslab/a.txt
0. exit
1. 240225163125          14bytes
2. 240225163207          11bytes
>> 0

% sudo ./ssu_backup recover ./a.txt -l
"/home/backup/240225163207/a.txt" recovered to "/home/oslab/a.txt"
```

그림 4-4-1. (전)/home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
│  ├ a.txt
│  └ b/
│     └ c.txt
├ a.txt
└ b/
```

그림 4-4-2. (전)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
│  └ a.txt
├ 240225163137/
│  └ c.txt
├ 240225163207/
│  └ a.txt
└ ssubak.log
```

그림 4-4-3. (후)/home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
│  ├ a.txt
│  └ b/
│     └ c.txt
├ a.txt
└ b/
```

그림 4-4-4. (후)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
│  └ a.txt
├ 240225163137/
│  └ c.txt
└ ssubak.log
```

- '-n' 옵션 입력 시에는 해당 옵션 뒤에 〈NEW_PATH〉을 필수로 입력받아, 백업 파일에 대해 새로운 경로를 지정하여 복구를 진행함. 만약 해당 경로가 없을 경우 디렉토리를 생성하여 경로를 설정함

예제 4-5-1. recover 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup recover /home/oslab/a -r -n ./new_d/1
"/home/backup/240225163125/a.txt" recovered to "/home/oslab/new_d/1/a.txt"
backup files of /home/oslab/a/c.txt
0. exit
1. 240225163125          14bytes
2. 240225163207          11bytes
>> 2
"/home/backup/240225163207/c.txt" recovered to "/home/oslab/new_d/1/c.txt"
"/home/backup/240225163125/b/c.txt" recovered to "/home/oslab/new_d/1/b/c.txt"
```

그림 4-5-1. (전)/home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
│  ├ a.txt
│  └ b/
│     └ c.txt
├ a.txt
└ b/
```

그림 4-5-2. (전)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
│  ├ a.txt
│  ├ b/
│  │  └ c.txt
│  └ c.txt
├ 240225163207/
│  └ c.txt
└ ssubak.log
```

그림 4-5-3. (후)/home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
│  ├ a.txt
│  └ b/
│     └ c.txt
├ a.txt
├ b/
└ new_d/
   └ 1/
      ├ a.txt
      ├ b/
      │  └ c.txt
      └ c.txt
```

그림 4-5-4. (후)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
│  └ c.txt
└ ssubak.log
```

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

- 경로를 입력하지 않을 경우, recover 명령어에 대한 Usage 출력 후 프로그램 비정상 종료(1점)
- 인자로 입력받은 경로가 길이 제한(4,096 Byte)를 넘거나 경로가 올바르지 않은 경우, 에러 처리 후 프로그램 비정상 종료(1점)
- 인자로 입력받은 경로와 일치하는 백업 파일이나 디렉토리가 없는 경우, 에러 처리 후 프로그램 비정상 종료(2점)
- 인자로 입력받은 경로가 사용자의 홈 디렉토리를 벗어날 경우, 에러 처리 후 프로그램 비정상 종료(2점)
- 인자로 입력받은 경로에 대한 백업 파일이 원본 파일과 내용이 같을 경우 복원을 진행하지 않고, ("원본 경로" not changed with "백업 경로")를 표준 출력(5점)
- 인자로 입력받은 옵션이 올바르지 않을 경우, backup 명령어에 대한 Usage 출력 후 프로그램 비정상 종료(1점)
- 인자로 입력받은 옵션 중 -d, -r을 사용하지 않았는데, 인자로 입력받은 경로가 존재하고 디렉토리일 경우, 에러 처리 후 프로그램 비정상 종료(2점)
- 인자로 입력받은 옵션 중 -d, -r을 사용하였는데, 인자로 입력받은 경로가 존재하고 파일일 경우, 에러 처리 후 프로그램 비정상 종료(2점)
- 인자로 입력받은 옵션 중 '-n' 뒤에 경로를 입력하지 않을 경우, 에러 처리 후 프로그램 비정상 종료(1점)
- 인자로 입력받은 옵션 중 '-n' 뒤에 경로가 올바르지 않을 경우, 에러 처리 후 프로그램 비정상 종료(1점)
- 인자로 입력받은 옵션 중 -d, -r을 사용하지 않았는데, '-n' 뒤에 경로가 존재하고 디렉토리일 경우, 혹은 인자로 입력받은 옵션 중 -d, -r을 사용하였는데, '-n' 뒤에 경로가 존재하고 파일일 경우, 에러 처리 후 프로그램 비정상 종료(1점)

5. 내장명령어 4. list

1) Usage : list [PATH]

- 백업 디렉토리 내의 파일 및 디렉토리 디렉토리 구조의 트리 형태로 출력하고 사용자 입력에 대해 명령을 수행함

2) 인자 설명

- 첫 번째 인자 [PATH]에 대해 해당 경로에 대해 재귀적으로 모든 백업 파일들을 트리 형태로 출력함. 첫 번째 인자는 생략 가능하며 생략 시 백업 디렉토리 내 모든 백업 파일들을 출력

3) 실행결과

- 첫 번째 인자 〈PATH〉에 대해 백업 파일들의 원본 경로를 기준으로 트리 형태로 출력함. 〈PATH〉 생략 시 백업 디렉토리 내 모든 백업 파일들을 기준으로 함.
- 백업 파일들의 트리를 출력할 때는 백업 파일들의 원본 경로들을 디렉토리 구조의 형태로 출력하며 위에서부터 번호를 매기며 한줄에는 원본 파일 및 디렉토리 이름, 파일일 경우 최근 백업 시간, 파일 크기(천 단위로 쉼표 출력)를 출력함. 이때 트리에 대한 출력 순서는 원본 파일 및 디렉토리 이름에 대해 오름차순으로 정렬함
- 목록을 출력한 후에는 사용자 입력을 기다리며 입력에 따라 명령을 실행함
- rm 〈INDEX〉 [OPTION]... 입력 시 해당 〈INDEX〉에 대한 경로를 remove 명령어와 같이 처리함
- rc 〈INDEX〉 [OPTION]... 입력 시 해당 〈INDEX〉에 대한 경로를 recover 명령어와 같이 처리함
- vi(m) 〈INDEX〉 입력 시 해당 〈INDEX〉에 대한 원본 파일에 대해 수정함
- exit 입력 시 프로그램 종료

예제 5. list 내장명령어 실행(현재 작업 디렉토리(pwd)가 "/home/oslab/" 일 때)

```
% sudo ./ssu_backup list a.txt
0. a.txt                    240225163148   1,025bytes
〉〉 exit

% sudo ./ssu_backup list a.txt
0. a.txt                    240225163148   1,025bytes
〉〉 vim 0
...

% sudo ./ssu_backup list
0. /home/oslab
1. ├ a/
2. │  ├ a.txt               240225163125   32bytes
3. │  ├ b/
4. │  │  └ c.txt            240225163125   14bytes
5. │  └ c.txt               240225163207   11bytes
6. └ a.txt                  240225163148   1,025bytes
〉〉 rm 5 -a
"/home/backup/240225163125/c.txt" removed by "/home/oslab/a/c.txt"
"/home/backup/240225163207/c.txt" removed by "/home/oslab/a/c.txt"

% sudo ./ssu_backup list ./a
0. a/
1. ├ a.txt                  240225163125   32bytes
2. └ b/
3.    └ c.txt               240225163125   14bytes
〉〉 rc 0 -r -n new_d
"/home/backup/240225163125/a.txt" recovered to "/home/oslab/new_d/a.txt"
"/home/backup/240225163125/b/c.txt" recovered to "/home/oslab/new_d/b/c.txt"

% sudo ./ssu_backup list
no backup file(s) in backup directory
```

그림 5-1. (전)/home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
│  ├ a.txt
│  └ b/
│     └ c.txt
└ b/
```

그림 5-3. (후)/home/oslab/ 디렉토리 트리 구조

```
/home/oslab/
├ a/
│  ├ a.txt
│  └ b/
│     └ c.txt
├ a.txt
├ b/
└ new_d/
   ├ a.txt
   └ b/
      └ c.txt
```

그림 5-2. (전)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163125/
│  ├ a.txt
│  ├ b/
│  │  └ c.txt
│  └ c.txt
├ 240225163137/
│  └ a.txt
├ 240225163148/
│  └ a.txt
├ 240225163207/
│  └ c.txt
└ ssubak.log
```

그림 5-4. (후)백업 디렉토리(/home/backup/) 트리 구조

```
/home/backup/
├ 240225163137/
│  └ a.txt
├ 240225163148/
│  └ a.txt
└ ssubak.log
```

4) 예외 처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)
- 잘못된 명령어 혹은 인덱스 입력 시, 에러 처리 후 프로그램 종료(1점)
- 인자로 입력받은 경로가 길이 제한(4,096 Byte)를 넘거나 경로가 올바르지 않은 경우, 에러 처리 후 프로그램 비정상 종료(1점)
- 인자로 입력받은 경로와 일치하는 백업 파일이나 디렉토리가 없는 경우, 에러 처리 후 프로그램 비정상 종료(1점)
- 인자로 입력받은 경로가 사용자의 홈 디렉토리를 벗어날 경우, 에러 처리 후 프로그램 비정상 종료(1점)

6. 내장명령어 5. 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 6. help 내장명령어 실행

```
% sudo ./ssu_backup help
Usage:
  〉backup 〈PATH〉 [OPTION]... : backup file if 〈PATH〉 is file
    -d : backup files in directory if 〈PATH〉 is directory
    -r : backup files in directory recursive if 〈PATH〉 is directory
    -y : backup file although already backuped
  〉remove 〈PATH〉 [OPTION]... : remove backuped file if 〈PATH〉 is file
    -d : remove backuped files in directory if 〈PATH〉 is directory
    -r : remove backuped files in directory recursive if 〈PATH〉 is directory
    -a : remove all backuped files
  〉recover 〈PATH〉 [OPTION]... : recover backuped file if 〈PATH〉 is file
    -d : recover backuped files in directory if 〈PATH〉 is directory
    -r : recover backuped files in directory recursive if 〈PATH〉 is directory
    -l : recover latest backuped file
    -n 〈NEW_PATH〉 : recover backuped file with new path
  〉list [PATH] : show backup list by directory structure
    〉〉 rm 〈INDEX〉 [OPTION]... : remove backuped files of [INDEX] with [OPTION]
    〉〉 rc 〈INDEX〉 [OPTION]... : recover backuped files of [INDEX] with [OPTION]
    〉〉 vi(m) 〈INDEX〉 : edit original file of [INDEX]
    〉〉 exit : exit program
  〉help [COMMAND] : show commands for program

% sudo ./ssu_backup help recover
Usage: recover 〈PATH〉 [OPTION]... : recover backuped file if 〈PATH〉 is file
  -d : recover backuped files in directory if 〈PATH〉 is directory
  -r : recover backuped files in directory recursive if 〈PATH〉 is directory
  -l : recover latest backuped file
  -n 〈NEW_PATH〉 : recover backuped file with new path
```

4) 예외 처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)
- 잘못된 내장명령어 입력 시 에러 처리 후 프로그램 비정상 종료(1점)

○ 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. ssu_backup, 2-4. 내장명령어 backup, remove, recover(각 옵션 및 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_backup - 10점

2. 내장명령어 1. backup - 25점
✓ '-d' 옵션 (5점)
✓ '-r' 옵션 (5점)
✓ '-y' 옵션 (3점)

3. 내장명령어 2. remove - 25점
✓ '-d' 옵션 (5점)
✓ '-r' 옵션 (5점)
✓ '-a' 옵션 (3점)

4. 내장명령어 3. recover - 25점
✓ '-d' 옵션 (5점)
✓ '-r' 옵션 (5점)
✓ '-l' 옵션 (3점)
✓ '-n' 옵션 (3점)

- 내장명령어 4. list - 7점
✓ 'rm' 입력 (2점)
✓ 'rc' 입력 (2점)
✓ 'vi(m)' 입력 (1점)
✓ 'exit' 입력 (1점)

- 내장명령어 5. help - 3점

- makefile 작성(매크로 사용하지 않아도 됨) - 5점