설계과제 1 개요 : SSU-Backup

○ 개요

- 리눅스 시스템 상에서 사용자가 백업을 원하는 파일이나 디렉토리를 옵션에 따라 추가, 삭제하고 백업된 파일을 다시 복구하는 것을 관리하는 프로그램

○ 목표

- 새로운 명령어를 시스템 함수를 사용하여 구현함으로써 쉘의 원리를 이해하고, 유닉스/리눅스 시스템에서 제공하는 여러 시스템 자료구조와 시스템콜 및 라이브러리를 이용하여 프로그램을 작성함으로써 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.
- 과제를 기한 내 새로 제출할 경우 기존 것은 삭제하지 않고 #P설계과제번호_학번_V1.0.zip 형태로 버전 이름만 붙이면 됨. 버전 이름은 대문자 V와 함께 integer를 1부터 incremental 증가시키면서 부여하면 됨. (예. V1, V2, V3, ...) 단, 처음 제출 시는 버전 번호를 붙이지 않아도 되며 두 번째부터 V1를 붙여 제출하면 됨. 단, #P설계과제번호_학번_V1.0.zip 압축 파일 내 구현보고서에는 버전 이름을 붙이지 않아도 됨.
- 제출한 압축 파일을 풀었을 때 해당 디렉토리에서 컴파일 및 실행이 되어야 함(특정한 디렉토리에서 실행해야 할 경우는 제외). 해당 디렉토리에서 컴파일이나 실행되지 않을 경우, 기본과제 및 설계과제 제출 방법(파일명, 디렉토리면, 컴파일 시에 포함되어야 할 파일 등)을 따르지 않으면 무조건 해당 과제 배점의 50% 감점
  - ✓ 설계과제의 기준 및 각 과제별 "필수기능요건"은 각 설계 과제 명세서에 별도 설명. 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)
- 기타 내용은 강의계획서 참고

○ 제출 기한

- 4월 3일(월) 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

- 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_backup 프로그램 실행 시 지정한 경로(/home/자신의 아이디의 홈디렉토리)에 백업디렉토리를 생성하고, 지정한 파일(정규 파일 또는 디렉토리만을 백업 대상으로 함. 재귀적으로 서브 디렉토리에 내에서도 정규파일과 디렉토리만 백업대상)을 백업디렉토리에 백업. 단 /home/자신의 아이디의 홈디렉토리/backup 디렉토리는 백업하지 않아야 함(재귀 호출로 문제 발생)
- 하나의 프로세스(단일 쓰레드)가 한 개 파일 또는 한 개 디렉토리(하부 디렉토리 및 파일 모두)를 백업해야 함
- ssu_backup은 파일 내용 비교를 위해 해시 함수(md5, sha1)를 사용하며 동일한(중복) 파일에 대해서 예외처리. “동일한(중복) 파일”은 파일의 해시값이 같은 정규 파일(텍스트, 바이너리 모두 포함)을 의미함
- ssu_backup에 내장명령어 중 모든 명령어는 프로세스를 생성(fork())하고 이를 실행(exec()류 함수)시켜줌 (exit 명령어는 exec() 류 함수 사용하지 않아도 됨)
- ssu_backup에 내장된 명령어들은 링크드 리스트로 파일 리스트를 관리해야 함

✓ 리눅스 상에서 파일 경로의 최대 크기는 4,096 바이트이며, 파일 이름의 최대 크기는 255 바이트임

- 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)

○ 설계 및 구현

1. ssu_backup

1) Usage : ssu_backup 〈md5 | sha1〉
- ssu_backup 실행 시 비교를 위한 해시 함수와 백업디렉토리를 생성할 경로를 인자로 입력
- ssu_backup 프로그램의 인자는 “해시 함수 (md5 / sha1)”임

2) 인자 설명
- 첫 번째 인자 md5와 sha1는 각각 md5와 sha1 해시 함수를 사용하는 것

3) 실행결과
- ssu_backup의 실행결과는 “학번〉”이 표준 출력되고 backup을 위한 내장명령어(add, remove, recover, ls, vi(m), help, exit) 입력 대기. 학번이 20230000일 경우 “20230000〉” 형태로 출력
- ssu_backup 프로그램 처음 실행 시 자신의 계정(알아서 마음대로 만들면 됨)의 홈디렉토리(예를 들어, 계정 아이디가 oslab일 경우, /home/oslab) 밑에 backup 디렉토리(/home/oslab/backup)를 생성함. 이미 backup 디렉토리가 존재할 시에는 생성하지 않음

예. ssu_backup 실행결과(현재 작업 디렉토리(pwd)는 /home/oslab/P1/으로 가정)

```
% ssu_backup md5
20230000〉

% ssu_backup
Usage: ssu_backup 〈md5 | sha1〉
```

예. ssu_backup 실행 후 프롬프트 상 명령어 입력

```
% ssu_backup sha1
20230000〉 ssu      //정의되지 않은 명령어 입력 시
Usage:
  〉 add [FILENAME] [OPTION]
     -d : add directory recursive
  〉 remove [FILENAME] [OPTION]
     -a : remove all file(recursive)
     -c : clear backup directory
  〉 recover [FILENAME] [OPTION]
     -d : recover directory recursive
     -n [NEWNAME] : recover file with new name
  〉 ls
  〉 vi
  〉 vim
  〉 help
  〉 exit

20230000〉 help
Usage:
  〉 add [FILENAME] [OPTION]
     -d : add directory recursive
  〉 remove [FILENAME] [OPTION]
     -a : remove all file(recursive)
     -c : clear backup directory
  〉 recover [FILENAME] [OPTION]
     -d : recover directory recursive
     -n [NEWNAME] : recover file with new name
  〉 ls
  〉 vi
  〉 vim
  〉 help
  〉 exit
```

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

- 첫 번째 인자가 없거나 해시(md5 / sha1)이 아닐 경우 또는 인자가 2개 이상인 경우 usage 출력 후 프로그램이 종료되어야 함(3점)
- 프롬프트 상에서 엔터만 입력 시 프롬프트 재출력(3점)
- 프롬프트 상에서 지정한 내장명령어 외 기타 명령어 입력 시 help 명령어의 결과를 출력 후 프롬프트 재출력(3점)

2. 내장명령어 1. add

1) Usage : add 〈FILENAME〉 [OPTION]

- 백업할 파일(FILENAME)을 입력받아 해당 파일 또는 디렉토리를 백업디렉토리에 백업

2) 인자 설명

- 첫 번째 인자 〈FILENAME〉은 백업할 파일이나 디렉토리의 상대경로와 절대경로 모두 입력 가능해야 함
- 두 번째 인자 [OPTION]은 '-d'만 있으며 생략 가능함(-d 옵션은 아래 설명 확인)

3) 실행결과

- 첫 번째 인자 〈FILENAME〉을 백업디렉토리에 백업 파일로써 저장함
- 백업 성공 시 지정한 백업디렉토리에 백업할 파일이 추가되었다고 출력("백업파일의 절대경로_백업시간(YYMMDDHHMMSS)" backuped)하고 프롬프트 재 출력
- 백업 성공 시 지정한 백업디렉토리에 백업할 파일의 이름은 기존 파일 이름 뒤에 백업시간("_YYMMDDHHMMSS")형태로 저장

그림 1. /home/oslab/ 디렉토리 트리 구조의 예

```
/home/oslab/
└ P1/
  ├ a/
  |  ├ a.txt
  |  ├ b/
  |  |  └ c.txt
  |  └ c.txt
  ├ a.txt
  └ b/
└ backup //백업디렉토리로 백업되면 안됨
```

예. add 내장명령어 실행(그림 1의 상황에서 현재 작업 디렉토리(pwd)가 “/home/oslab/P1/” 일 때)

```
20230000〉 add
Usage : add 〈FILENAME〉 [OPTION]

20230000〉 add a.txt
“/home/oslab/backup/P1/a.txt_230227172231” backuped

20230000〉 add /home/oslab/P1/a.txt
“/home/oslab/backup/P1/a.txt_230227172233” backuped

20230000〉 add a/a.txt
“/home/oslab/backup/P1/a/a.txt_230227172236” backuped

20230000〉 add /home/oslab/P1/a       //P1/a이 디렉토리인 경우, 예외 처리 5번째 확인
“/home/oslab/P1/a” is a directory file

20230000〉 add /home/asd       //경로가 사용자 홈 디렉토리를 벗어난 경우, 예외 처리 3번째 확인
“/home/asd” can't be backuped

20230000〉 add /home/oslab/backup/a.txt       //경로가 백업디렉토리를 포함할 경우, 예외 처리 3번째 확인
“/home/oslab/backup/a.txt” can't be backuped
```

- ‘-d’ 옵션 입력시에는 〈FILENAME〉가 디렉토리일 시 해당 경로 아래 있는 모든 파일들에 대해 백업을 진행함. 이 때, 서브 디렉토리 내에 모든 파일들에 대해서도 재귀적으로 탐색하여 백업을 진행함

그림 2. /home/oslab/ 디렉토리 트리 구조의 예

```
/home/oslab/
└ P1/
  ├ a.txt
  ├ b.jpg
  ├ D1/
  |  └ b.jpg
  └ D2/
```

예. add 내장명령어의 -d 옵션(그림 1의 상황에서 현재 작업 디렉토리(pwd)가 "/home/oslab/P1/" 일 때)
20230000〉 add ./ -d // 현재 작업 디렉토리("/home/oslab/P1/")에 대하여 백업할 경우(D2 디렉토리에는 파일이 없기 때문에 백업하지 않음) "/home/oslab/backup/P1/a.txt_230227172231" backuped "/home/oslab/backup/P1/b.jpg_230227172231" backuped "/home/oslab/backup/P1/D1/b.jpg_230227172331" backuped 20230000〉 add /home/oslab/P1 -d // 그림 2의 예에서 백업디렉토리(/home/oslab/backup/)에 D1 디렉토리에 있는 b.jpg가 /home/oslab/backup 디렉토리에 /home/oslab/backup/P1/D1/b.jpg만 이미 존재(백업)하고 있는 경우. case1} 백업할 /home/oslab/P1/D1/b.jpg와 백업되어 있는 /home/oslab/backup/P1/D1/b.jpg가 완전히 동일할 경우, case2} 백업할 /home/oslab/P1/D1/b.jpg와 백업되어 있는 /home/oslab/backup/P1/D1/b.jpg가 완전히 동일하지 않을 경우 case1의 결과 "/home/oslab/backup/P1/a.txt_230227172233" backuped "/home/oslab/backup/P1/b.jpg_230227172231" is already backuped "/home/oslab/backup/P1/D1/b.jpg_230227172333" backuped case2의 결과 "/home/oslab/backup/P1/a.txt_230227172233" backuped "/home/oslab/backup/P1/b.jpg_230227172233" backuped //백업 시간이 다르게 저장 "/home/oslab/backup/P1/D1/b.jpg_230227172333" backuped

4) 예외 처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)
- 첫 번째 인자 입력이 없거나 올바르지 않은 경로(해당 파일이나 디렉토리 없을 경우 포함) 경우 포함 입력 시 Usage 출력 후 프롬프트 재출력(2점)
- 첫 번째 인자로 입력받은 경로(절대 경로)가 길이 제한(255 Byte)을 넘거나 해당 경로가 일반 파일이나 디렉토리가 아니거나 접근권한이 없는 경우 에러 처리 후 프롬프트 재출력(2점)
- 첫 번째 인자로 입력받은 경로(절대 경로)가 사용자의 홈 디렉토리($HOME 또는 ~)를 벗어나거나 백업 디렉토리 혹은 백업 디렉토리 내 파일 및 디렉토리를 포함한다면 "〈입력받은 경로〉 can't be backuped" 표준 출력 후 프롬프트 재출력(2점)
- 두 번째 인자로 올바르지 않은 옵션이 들어왔을 경우 Usage 출력 후 프롬프트 재출력(2점)
- 첫 번째 인자가 디렉토리일 경우 '-d' 옵션을 사용하지 않았다면 에러 처리 후 프롬프트 재출력(2점)
- 백업파일의 기존 경로가 같고, 해시값이 같은 파일이 이미 백업디렉토리에 존재한다면 백업을 하지 않고 "〈백업파일의 절대경로〉 is already backuped" 표준 출력 후 프롬프트 재출력(2점)

3. 내장명령어 2. remove

1) Usage : remove 〈FILENAME〉 [OPTION]
- 백업된 파일(FILENAME)을 입력받아 해당 파일 또는 디렉토리를 백업디렉토리(디폴트 백업디렉토리는 /home/backup임)에서 삭제

2) 인자 설명
- 첫 번째 인자 〈FILENAME〉은 백업 디렉토리 내 백업된 파일이나 디렉토리의 상대경로와 절대경로 모두 입력 가능해야함
- 두 번째 인자 [OPTION]은 '-a'와 '-c'가 있으며 모두 생략 가능함(-a와 -c 옵션은 아래 설명 확인)

3) 실행결과
- 첫 번째 인자 〈FILENAME〉에 대해 백업디렉토리에 있는 백업 파일(정규파일만, 디렉토리는 -a 옵션 처리) 중 인자로 주어진 파일의 경로가 〈FILENAME〉과 일치하는 백업파일을 삭제함. 만약 백업파일이 여러개 존재 시 "backup file list of "〈원본 파일 경로〉"" 출력 후 다음 줄부터 리스트로 백업 파일의 백업 시간과 크기(천 단위로 쉼표 출력)를 출력하고 사용자 입력 대기
- 첫 번째 인자로 상대경로를 입력했을 경우 입력한 상대경로에 대한 백업파일을 찾아야 하기 때문에 첫 번째 인자로 입력받은 상대경로를 백업디렉토리에 맞게 변환해주는 과정 필요
- 사용자가 입력한 번호에 맞는 백업파일을 삭제 진행 후 결과 출력 및 프롬프트 재출력. 0번 입력시 명령어 실행 종료 후 프롬프트 재출력

그림 3. /home/oslab/backup/ 디렉토리 트리 구조의 예

```
/home/oslab/backup/
└ P1/
  ├ a/
  |  ├ a.txt_20322172302
  |  ├ a.txt_20322172336
  |  ├ b/
  |  |  └ c.txt_20322172302
  |  ├ b.txt_190315124532
  |  ├ b.txt_230116132511
  |  ├ d.txt_20322182157
  |  ├ d.txt_20322193011
  |  └ d.txt_20322210423
  ├ a.txt_203227172231
  └ a.txt_203227172236
```

그림 4. /home/oslab/ 디렉토리 트리 구조의 예

```
/home/oslab/
└ P1/
  ├ a/
  |  ├ a.txt
  |  └ b/
  |     └ c.txt
  └ a.txt
└ backup/
```

예. remove 내장명령어 실행(그림 3과 4의 상황에서 현재 작업 디렉토리(pwd)가 “/home/oslab/P1/” 일 때)

```
20230000〉 remove
Usage : remove 〈FILENAME〉 [OPTION]

20230000〉 remove a.txt
backup file list of "/home/oslab/P1/a.txt" // 동일한 파일이 2개 있는 경우
0. exit
1. 230227172231          13bytes
2. 230227172236          15bytes
Choose file to remove
〉〉 1
“/home/oslab/backup/P1/a.txt_230227172231” backup file removed

20230000〉 remove /home/oslab/P1/a/b.txt
backup file list of "/home/oslab/P1/a/b.txt" // 동일한 파일이 2개 있는 경우
0. exit
1. 190315124532          35bytes
2. 230116132511          34bytes
Choose file to remove
〉〉 1
“/home/oslab/backup/P1/a/b.txt_190315124532” backup file removed

20230000〉 remove a/d.txt  //백업하려는 파일이 사용자의 홈 디렉토리에 없지만 백업디렉토리에는 백업파일이 있는 경우
backup file list of "/home/oslab/P1/a/d.txt"
0. exit
1. 230227182157    361bytes
2. 230227193011    512bytes
3. 230227210423    1,003bytes
Choose file to remove
〉〉 3
“/home/oslab/backup/P1/a/d.txt_230227210423” backup file removed
```

- ‘-a’ 옵션 입력 시에는 기존경로가 〈FILENAME〉과 일치하는 모든 백업파일을 삭제함. 이 때 〈FILENAME〉이 디렉토리라면 기존경로가 〈FILENAME〉 하위 파일인 백업 파일을 모두 삭제함

그림 5. /home/oslab/backup/ 디렉토리 트리 구조의 예

```
/home/oslab/backup/
└ P1/
  ├ a/
  |  ├ a.txt_20322172302
  |  ├ b.txt_20322172302
  |  ├ b/
  |  |  └ c.txt_20322172302
  |  └ c.txt_20322172302
  └ a.txt_203227172231
```

예. remove 내장명령어의 -a 옵션(그림 5의 상황에서 현재 작업 디렉토리(pwd)가 “/home/oslab/P1/” 일 때)
20230000〉 remove a -a “/home/oslab/backup/P1/a/a.txt_230227172302” backup file removed “/home/oslab/backup/P1/a/b.txt_230227172302” backup file removed “/home/oslab/backup/P1/a/c.txt_230227172302” backup file removed “/home/oslab/backup/P1/a/b/c.txt_230227172302” backup file removed 20230000〉 remove /home/oslab/P1/a //절대경로 입력 시 위와 동일 “/home/oslab/backup/P1/a/a.txt_230227172302” backup file removed “/home/oslab/backup/P1/a/b.txt_230227172302” backup file removed “/home/oslab/backup/P1/a/c.txt_230227172302” backup file removed “/home/oslab/backup/P1/a/b/c.txt_230227172302” backup file removed

- ‘-c’ 옵션 입력시에는 백업디렉토리 내에 있는 모든 백업 파일을 삭제함. 삭제 후 “backup directory clear” 출력 후 프롬프트 재출력
- /home/backup 내 서브디렉토리를 포함해서 삭제된 모든 백업된 파일과 모든 서브디렉토리를 삭제하고 서브 디렉토리 내 모든 파일 수 + 모든 서브 디렉토리 수를 명시. 단, /home/backup 디렉토리는 삭제되면 안됨.

그림 6. /home/oslab/backup/ 디렉토리 트리 구조의 예
/home/oslab/backup/ └ P1/ ├ a/ | ├ a.txt_20322172302 | └ b/ | └ c.txt_20322172302 └ a.txt_203227172231

예. remove 내장명령어의 -c 옵션(그림 6의 상황에서 현재 작업 디렉토리(pwd)가 “/home/oslab/P1/” 일 때)
20230000〉 remove -c backup directory cleared(3 regular files and 3 subdirectories totally). // 그림 4의 디렉토리의 구조상 P1, P1/a/, P1/a/b/ 등 3개 디렉토리와 P1/a.txt_, P1/a/a/txt_, P1/a/b/c.txt_ 등 3개의 정규파일이 백업되어 있을 경우 20230000〉 remove -c no file(s) in the backup // 백업 디렉토리에 아무 파일도 없을 경우, 예외 처리 7번째 확인

4) 예외 처리(미 구현 시 아래 점수만큼 감점, 필수 기능 구현 여부 판단과는 상관 없음)
- 첫 번째 인자 입력이 없거나 올바르지 않은 경로(해당 백업 파일이나 디렉토리 없을 경우 포함) 입력 시 Usage 출력 후 프롬프트 재출력(3점)
- 첫 번째 인자로 입력받은 경로(절대 경로)가 길이 제한(255 Byte)을 넘거나 해당 경로가 일반 파일이나 디렉토리가 아니거나 접근권한이 없는 경우 에러 처리 후 프롬프트 재출력(5점)
- 첫 번째 인자로 입력받은 경로(절대 경로)가 사용자의 홈 디렉토리($HOME 또는 ~)를 벗어나거나 백업 디렉토리 혹은 백업 디렉토리 내 파일 및 디렉토리를 포함한다면 “〈입력받은 경로〉 can’t be backuped” 표준 출력 후 프롬프트 재출력(5점)
- 첫 번째 인자가 디렉토리인 경우 ‘-a’ 옵션을 사용하지 않았다면 에러 처리 후 프롬프트 재출력(3점)
- 두 번째 인자로 올바르지 않은 옵션이 들어왔거나 ‘-a’ 옵션과 ‘-c’ 옵션이 동시에 들어왔을 경우 Usage 출력 후 프롬프트 재출력(3점)
- ‘-c’ 옵션 사용 시 다른 인자를 입력하면 Usage 출력 후 프롬프트 재출력(2점)
- ‘-c’ 옵션 사용 시 백업디렉토리에 아무 파일이 없을 경우 “no file(s) in the backup” 출력 후 프롬프트 재출력(2점)

4. 내장명령어 3. recover

1) Usage : recover 〈FILENAME〉 [OPTION]
- 백업된 파일(FILENAME)을 입력받아 해당 파일 또는 디렉토리를 백업디렉토리(디폴트 백업디렉토리는 /home/backup 임)에서 파일 복구 진행

2) 인자 설명
- 첫 번째 인자 〈FILENAME〉은 백업 디렉토리 내 백업된 파일이나 디렉토리의 상대경로와 절대경로 모두 입력 가능해야함
- 두 번째 인자 [OPTION]은 ‘-d’와 ‘-n’가 있으며 모두 생략 가능함(-d와 -n 옵션은 아래 설명 확인)

3) 실행결과

- 첫 번째 인자 〈FILENAME〉에 대해 백업디렉토리에 있는 백업 파일(정규파일만, 디렉토리는 -d 옵션 처리) 중 인자로 주어진 파일의 경로가 〈FILENAME〉과 일치하는 백업파일을 복구함. 만약 백업파일이 여러개 존재 시 “backup file list of “〈원본 파일 경로〉”” 출력 후 다음 줄부터 리스트로 백업 파일의 백업 시간과 크기(천 단위로 쉼표 출력)를 출력하고 사용자 입력 대기
- 첫 번째 인자로 상대경로를 입력했을 경우 입력한 상대경로에 대한 백업파일을 찾아야 하기 때문에 첫 번째 인자로 입력받은 상대경로를 백업디렉토리에 맞게 변환해주는 과정 필요
- 사용자가 입력한 번호에 맞는 백업파일을 삭제 및 기존 파일에 덮어쓰기 진행(기존 파일이 삭제되었다면 생성) 후 결과 출력 및 프롬프트 재출력. 0번 입력시 명령어 실행 종료 후 프롬프트 재출력

그림 7. /home/oslab/backup/ 디렉토리 트리 구조의 예

```
/home/oslab/backup/
└ P1/
  ├ a/
  │ ├ a.txt_20322172302
  │ ├ b/
  │ │ └ c.txt_20322172302
  │ ├ d.txt_20322182157
  │ ├ d.txt_20322193011
  │ └ d.txt_20322210423
  ├ a.txt_203227172231
  └ a.txt_203227172236
```

그림 8. /home/oslab/ 디렉토리 트리 구조의 예

```
/home/oslab/
└ P1/
  ├ a/
  │ ├ a.txt
  │ └ b/
  │   └ c.txt
  └ a.txt
└ backup/
```

예. recover 내장명령어 실행(그림 7과 8의 상황에서 현재 작업 디렉토리(pwd)가 “/home/oslab/P1/” 일 때)

```
20230000〉 recover
Usage : recover 〈FILENAME〉 [OPTION]

20230000〉 recover a.txt
backup file list of "/home/oslab/P1/a.txt"
0. exit
1. 230227172231                13bytes
2. 230227172236                15bytes
Choose file to recover
〉〉 2
“/home/oslab/backup/P1/a.txt_230227172236” backup recover to “/home/oslab/P1/a.txt”

20230000〉 recover a/d.txt  //백업 복구하려는 파일이 사용자의 홈 디렉토리에 없지만 백업디렉토리에는 백업파일이 있는 경우
backup file list of "/home/oslab/P1/a/d.txt"
0. exit
1. 230227182157    361bytes
2. 230227193011    512bytes
3. 230227210423    1,003bytes
Choose file to recover
〉〉 3
“/home/oslab/backup/P1/a/d.txt_230227210423” backup file recover
```

- ‘-d’ 옵션 입력 시에는 기존경로가 〈FILENAME〉인 디렉토리의 모든 백업파일과 서브 디렉토리 내부의 파일들까지 재귀적으로 복구함

그림 9. /home/oslab/backup/ 디렉토리 트리 구조의 예

```
/home/oslab/backup/
└ P1/
  ├ a/
  │ ├ a.txt_20322172302
  │ ├ b.txt_20322172302
  │ ├ b/
  │ │ └ c.txt_20322172302
  │ └ c.txt_20322172302
  └ a.txt_203227172231
```

그림 10. /home/oslab/ 디렉토리 트리 구조의 예

```
/home/oslab/
└ P1/
  ├ a/
  │ ├ a.txt
  │ └ b/
  │   └ c.txt
  └ a.txt
└ backup/
```

예. recover 내장명령어의 -d 옵션(그림 9와 10의 상황에서 현재 작업 디렉토리(pwd)가 “/home/oslab/P1/” 일 때)

```
20230000〉 recover a -d  // a는 디렉토리임
“/home/oslab/backup/P1/a/a.txt_230227172302” backup recover to “/home/oslab/P1/a/a.txt”
“/home/oslab/backup/P1/a/b.txt_230227172302” backup recover to “/home/oslab/P1/a/b.txt”
“/home/oslab/backup/P1/a/c.txt_230227172302” backup recover to “/home/oslab/P1/a/c.txt”
“/home/oslab/backup/P1/a/b/c.txt_230227172302” backup recover to “/home/oslab/P1/a/b/c.txt”

20230000〉 recover /home/oslab/P1/a -d  //백업디렉토리(그림 9) 내 P1/a 디렉토리 내 b.txt에 대한 백업 파일이 2개일 경우
“/home/oslab/backup/P1/a/a.txt_230227172302” backup recover to “/home/oslab/P1/a/a.txt”
backup file list of "/home/oslab/P1/a/b.txt"
0. exit
1. 230227172302          13bytes
2. 230227172413          15bytes
Choose file to recover
〉〉 2
“/home/oslab/backup/P1/a/b.txt_230227172413” backup recover to “/home/oslab/P1/a/b.txt”
“/home/oslab/backup/P1/a/c.txt_230227172302” backup recover to “/home/oslab/P1/a/c.txt”
“/home/oslab/backup/P1/a/b/c.txt_230227172302” backup recover to “/home/oslab/P1/a/b/c.txt”
```

- ‘-n 〈NEWFILE〉’ 옵션 입력 시에는 기존경로가 〈FILENAME〉과 일치하는 백업파일을 〈NEWFILE〉 경로로 바꾸어 복구를 진행하고 〈NEWFILE〉의 파일/디렉토리가 없다면 생성 후 덮어씀. ‘-d’ 입력과 동시에 사용할 시에는 기존경로가 〈FILENAME〉인 디렉토리의 모든 백업파일을 〈NEWFILE〉 경로로 복구를 진행.

그림 11. /home/oslab/backup/ 디렉토리 트리 구조의 예

```
/home/oslab/backup/
└ P1/
  ├ a/
  |  ├ a.txt_20322172302
  |  ├ b.txt_20322172302
  |  ├ b/
  |  |  └ c.txt_20322172302
  |  └ c.txt_20322172302
  ├ a.txt_203227172231
  └ a.txt_203227172236
```

그림 12. /home/oslab/ 디렉토리 트리 구조의 예

```
/home/oslab/
└ P1/
  ├ a/
  |  ├ a.txt
  |  └ b/
  |     └ c.txt
  └ a.txt
└ backup/
```

예. recover 내장명령어의 -n 옵션(그림 11과 12의 상황에서 현재 작업 디렉토리(pwd)가 “/home/oslab/P1/” 일 때)

```
20230000〉 recover a.txt -n new_a.txt
backup file list of "/home/oslab/P1/a.txt"
0. exit
1. 230227172231          13bytes
2. 230227172236          15bytes
Choose file to recover
〉〉 1
“/home/oslab/backup/P1/a.txt_230227172231” backup recover to “/home/oslab/P1/new_a.txt”
```

예. recover 내장명령어의 -d, -n 옵션(그림 11과 12의 상황에서 현재 작업 디렉토리(pwd)가 “/home/oslab/P1/” 일 때)

```
20230000〉 recover a -d -n new_a
“/home/oslab/backup/a.txt_230227172302” backup recover to “/home/oslab/P1/new_a/a.txt”
“/home/oslab/backup/b.txt_230227172302” backup recover to “/home/oslab/P1/new_a/b.txt”
“/home/oslab/backup/c.txt_230227172302” backup recover to “/home/oslab/P1/new_a/c.txt”
“/home/oslab/backup/c.txt_230227172302” backup recover to “/home/oslab/P1/new_a/b/c.txt”
```

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

- 첫 번째 인자 입력이 없거나 올바르지 않은 경로(해당 백업 파일이나 디렉토리 없을 경우 포함) 입력 시 Usage 출력 후 프롬프트 재출력(3점)
- 첫 번째 인자로 입력받은 경로(절대 경로) 혹은 ‘-n’ 옵션 뒤에 〈NEWFILE〉 인자가 길이 제한(255 Byte)을 넘거나 해당 경로가 일반 파일이나 디렉토리가 아니거나 접근권한이 없는 경우 에러 처리 후 프롬프트 재출력(5점)

- 첫 번째 인자로 입력받은 경로(절대 경로) 혹은 '-n' 옵션 뒤에 〈NEWFILE〉 인자가 사용자의 홈 디렉토리(%HOME 또는 ~)를 벗어나거나 백업 디렉토리 혹은 백업 디렉토리 내 파일 및 디렉토리를 포함한다면 "〈입력받은 경로〉 can't be backuped" 표준 출력 후 프롬프트 재출력(5점)
- 첫 번째 인자가 디렉토리일 경우 '-d' 옵션을 사용하지 않았다면 에러 처리 후 프롬프트 재출력(3점)
- 두 번째 인자로 올바르지 않은 옵션이 들어왔을 경우 Usage 출력 후 프롬프트 재출력(3점)
- 백업파일의 기존 경로가 〈FILENAME〉과 같고, 해당 파일과 해시값이 같다면 백업 복구를 진행하지 않음(5점)
- 기존 경로에 파일이 없다면 백업파일 상대로 생성(3점)

5. 내장명령어 4. ls, vi(m)

1) Usage : 기존 시스템 함수 ls, vi(m)과 동일
- exec() 류 함수를 이용하여 시스템 함수 ls, vi(m)을 실행(system() 함수 사용 불가)

2) 인자 설명
- 기존 시스템 함수 ls, vi(m)과 동일

3) 실행결과
- 프로그램 실행 경로를 기준으로 기존 시스템 함수 ls, vi(m)을 실행

예. list 실행 시 표준 출력

```
20230000〉 ls
Makefile   a   a.txt   b.txt   ssu_backup
```

6. 내장명령어 5. help

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

2) 실행결과
- 프로그램 내장명령어 help 실행

예. help 내장명령어 실행

```
20230000〉 help
Usage:
  〉 add [FILENAME] [OPTION]
    -d : add directory recursive
  〉 remove [FILENAME] [OPTION]
    -a : remove all file(recursive)
    -c : clear backup directory
  〉 recover [FILENAME] [OPTION]
    -d : recover directory recursive
    -n [NEWNAME] : recover file with new name
  〉 ls
  〉 vi
  〉 vim
  〉 help
  〉 exit
```

7. 내장명령어 6. exit

1) Usage : exit
- 현재 실행중인 ssu_backup 프로그램 종료

2) 실행결과
- 프로그램 종료

예. exit 실행 시 표준 출력

```
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 : 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 및 SHA1

✓ 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

✓ SHA1 해시값을 구하기 위해 SHA1(openssl/sha.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
- SHA1 관련 함수를 사용하기 위해 컴파일 시 "-lcrypto" 옵션 필요
- SHA1() 사용법은 https://www.openssl.org/docs/man1.1.1/man3/SHA1.html 및 https://github.com/Chronic-Dev/openssl/blob/master/crypto/sha/sha1.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. 내장명령어 add, recover(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. add - 15점
   ✓ '-d' 옵션
3. 내장명령어 2. remove - 30점
   ✓ '-a' 옵션
   ✓ '-c' 옵션
4. 내장명령어 3. recover - 30점
   ✓ '-d' 옵션
   ✓ '-n' 옵션

- 내장명령어 4. ls, vi(m) - 5점
- 내장명령어 5. help - 3점
- 내장명령어 6. exit - 2점
- makefile 작성(매크로 사용하지 않아도 됨) - 5점