

설계과제 개요 1 : SSU-Backup

○ 개요

- 리눅스 시스템 상에서 사용자가 백업을 원하는 파일이나 디렉토리를 옵션에 따라 추가 삭제하고 백업된 파일을 다시 복구하, 는 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 향상 시킴

○ 팀 구성

- 개인별 프로젝트

○ 보고서 제출 방법

- 설계과제는 “**#P _ .zip 설계과제번호 학번**”(. #P1_20140000_V1.zip) classroom.google.com 에 형태로 압축하여 에 제출해야 함.
- “**#P _ .zip** 설계과제번호 학번 내 보고서인 설계과제번호 와 헤더파일 및 소스코드 등 해당 디렉토리에서 컴파일과 ”**#P .hwp**” 실행이 가능하도록 모든 파일 등 컴파일하고 실행하기 위한 파일들 을 포함시켜야 함 단 특정한 (makefile, obj, *.c, *.h) . . 디렉토리에서 실행해야 할 경우는 예외.
- 구현보고서인 설계과제번호 에는 과제개요 명세에서 주어진 개요를 그대로 쓰면 안됨 자기가 구현한 내용 요약 “**#P .hwp**”
1. (.) 2. (,) 3. (, ,) 4. (기능 구현한 기능 요약 상세설계 함수 및 모듈 구성 순서도 구현한 함수 프로토타입 등 실행결과 구현한 모든 기능 및 실행 결과 캡처 를 반드시 포함시켜야 함) .
- 과제를 기한 내 새로 제출할 경우 기존 것은 삭제하지 않고 설계과제번호 학번 형태로 버전 이름만 붙이면 됨 **#P _ _V1.0.zip** . 버전 이름은 대문자 와 함께 를 부터 증가시키면서 부여하면 됨 예 단 처음 **V integer 1 incremental** . (. V1, V2, V3, ...) , 제출 시는 버전 번호를 붙이지 않아도 되며 두 번째부터 를 붙여 제출하면 됨 단 설계과제번호 학번 압축 **V1 . , #P _ _V1.0.zip** 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 제 (외 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 감점) **50%**
✓ 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 설계과제번호 점 개요 점 기능 점 상세설계 점 실행 결과 점 **#P .hwp (15) : 1 , 1 , 10 , 3**
✓ 소스코드 점 소스코드 주 **(85) : 5** , 석 점 실행 여부 점 설계 요구에 따르지 않고 설계된 경우 소스코드 주 **80** (석 및 실행 여 부는 점 부여 설계 요구에 따라 설계된 경우 기능 **0 . 100** 미구현 부분을 설계명세서의 점 기준에서 해당 기능 감점 후 이를 **80**) 점으로 환산
- 기타 내용은 강의계획서 참고

○ 제출 기한

- **4 3 () 11 5** 월 일 월 오후 시 **9분 59초**

○ 보고서 및 소스코드 배점

- 보고서는 다음과 같은 양식으로 작성(강의계획서 **FAQ**) 참고

- | |
|--|
| <ol style="list-style-type: none">1. (1) // 과제 개요 점 명세에 주어진 개요를 더 상세하게 작성2. (1) // 구현 기능 점 함수 프로토타입 반드시 포함3. (10) // () 상세 설계 점 함수 기능별 흐름도 순서도 반드시 포함4. (3) // 실행결과 점 테스트 프로그램의 실행결과 캡처 및 분석 |
|--|

- 소스코드 및 실행 여부 점 주 **(85) // (5)** , (석 점 실행 여부 점 **80**)

○ ssu_backup 프로그램 기본 사항

- 리눅스 시스템에서 생성한 자신의 사용자 아이디에 대해 사용자 홈 디렉토리를 확인 예 자 (. 신의 리눅스 시스템상 계정 아이디가 일 경우는 oslab \$HOME /home/oslab . (임 또 %echo \$HOME, 는 파일에서 해당 아이디의 %cat /etc/passwd 홈 디렉토리 확인)
- ssu_backup (/home/) , 프로그램 실행 시 지정한 경로 자신의 아이디의 홈디렉토리 에 백업디렉토리를 생성하고 지정한 파일 (.) 정규 파일 또는 디렉토리만을 백업 대상으로 함 재귀적으로 서브 디렉토리에 내에서도 정규파일과 디렉토리만 백업대상을 백업디렉토리에 백업 단 자 . /home/ /backup (신의 아이디의 홈디렉토리 디렉토리는 백업하지 않아야 함 재귀 호출로 문제 발생)
- 하나의 프로세스 단일 쓰 () () 레드 가 한 개 파일 또는 한 개 디렉토리 하부 디렉토리 및 파일 모두 를 백업해야 함
- ssu_backup (m 은 파일 내용 비교를 위해 해시 함수 를 사용하며 d5, sha1) () . 동일한 중복 파일에 대해서 예외처리 “동일한 () 중복 파일 은 파일의 해시 ” 값이 같은 정규 파일 (,) 텍스트 바이너리 모두 포함 을 의미함
- ssu_backup (fork()) (e 에 내장명령어 중 모든 명령어는 프로세스를 생성 하고 이를 실행 xec()) e 류 함수 시켜줌 (xit 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 , 255 바이트이며 파일 이름의 최대 크기는 바이트임 - 프로그램 전체에서 system() . 0 절대 사용 불가 사용 시 점 처리
- getopt() 라이브러리를 사용하여 옵션 처리 권장
- ssu_backup foreground 은 로만 수행되는 것으로 가정함 수행은 안되는 것으로 함 d (& background)

○ 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 와 는 md5 sha1 m 각각 와 해시 함수를 사용하는 것 d5 sha1

3) 실행결과

- ssu_backup의 실행결과는 학번 이 표 “>” 준 출력되고 을 위한 내 backup (a 장명령어 dd, remove, recover, ls, vi(m), help, exit) . 20230000 입력 대기 학번이 일 경우 형태로 출력 “20230000>”
- ssu_backup 프로그램 처음 실행 시 자신의 계정의 홈디렉토리(oslab , /home/oslab) 계정 아이디가 일 경우 backu 밑에 p 디렉토리(/home/oslab/backup)를 생성함 이. backup 미 디렉토리가 존재할 시에는 생성하지 않음

예 실행결과 현 . ssu_backup (pw 재 작업 디렉토리는 으로 가정 d) /home/oslab/P1/)

% ssu_backup md5
20230000>

% ssu_backup
Usage: ssu_backup <md5 | sha1>

예 실행 . ssu_backup 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 이 아 (md5 / sha1) 2 usage 닐 경우 또는 인자가 개 이상인 경우 출력 후 프로그램이 종료되어야 함 점(3)
- 프롬프트 상에서 엔터만 입력 시 프롬프트 재출력 점(3)
- 프롬프트 상에서 지정한 내장명령어 외 기타 명령어 입력 시 명령어의 결과를 출력 help (3) 후 프롬프트 재출력 점

2. 내장명령어 1. add

1) Usage : add <FILENAME> [OPTION]

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

2) 인자 설명

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

3) 실행결과

- 첫 번째 인자 을 백업디렉토리에 백업 파일로써 <FILENAME> 저장함
- 백업 성공 시 지정한 백업디렉토리에 백업할 파일이 추가되었다고 출력 백업파일의 (“ 절대경로 백업시 _ (간 YYMMDDHHMMS S)” backedup)하고 프롬프트 재 출력
- 백업 성공 시 지정한 백업디렉토리에 백업할 파일의 이름은 기존 파일 이름 뒤에 백업시간(“_YYMMDDHHMMS”)형태로 저장

그림 디렉토리 트리 구조의 예 1. /home/oslab/

```
/home/oslab/
└─ P1/
  a/ └─
  a.t | └─ xt
  b/ | └─
  c.t | | └─ xt
  c.t | └─ xt
  a.t └─ xt
  b/ └─
    └─ backup //백업디렉토리로 백업되면 안됨
```

예 내 . add (1 (pw 장명령어 실행 그림 의 상황에서 현재 작업 디렉토리 가 일 때 d) “/home/oslab/P1/”)

20230000> add

Usage : add <FILENAME> [OPTION]

-d : add directory recursive

20230000> add a.txt

“/home/oslab/backup/P1/a.txt_230227172231” backupped

20230000> add /home/oslab/P1/a.txt

“/home/oslab/backup/P1/a.txt_230227172233” backupped

20230000> add a/a.txt

“/home/oslab/backup/P1/a/a.txt_230227172236” backupped

20230000> add /home/oslab/P1/a //P1/a , 5 이 디렉토리인 경우 예외 처리 번째 확인

“/home/oslab/P1/a” is a directory file

20230000> add /home/asd // , 3 경로가 사용자 홈 디렉토리를 벗어난 경우 예외 처리 번째

확인 “/home/asd” can't be backupped

20230000> add /home/oslab/backup/a.txt // , 3 경로가 백업디렉토리를 포함할 경우 예외 처리 번째 확인

“/home/oslab/backup/a.txt” can't be backupped

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

그림 디렉토리 트리 구조의 예 2. /home/oslab/

```
/home/oslab/
└─ P1/
  a.t └─ xt
  b.jpg └─
    └─ D1/
      b.jpg | └─
        └─ D2/
```

예 내 . add 장명령어의 옵션 그 -d (1 (pw 링 의 상황에서 현재 작업 디렉토리 가 일 때 d) "/home/oslab/P1/")

20230000> add ./ -d // (현재 작업 디렉토리 에 대하여 백업할 경우 디렉토리에는 파일이 "/home/oslab/P1/") (D2 없 기 때문에 백업하지 않음)

"/home/oslab/backup/P1/a.txt_230227172231" backedup

"/home/oslab/backup/P1/b.jpg_230227172231" backedup

"/home/oslab/backup/P1/D1/b.jpg_230227172231" backedup

20230000> add /home/oslab/P1 -d // 그림 의 예에서 백업디렉토리 에 디렉토리에 2 (/home/oslab/backup/) P1 있는 b.jpg /home/oslab/backup /home/oslab/backup/P1/b.jpg () . case1 가 디렉토리에 만 이미 존재 백업 하고 있는 경우 백업할 } /home/oslab/P1/b.jpg /home/oslab/backup/P1/b.jpg , case2 와 백업되어 있는 가 완전히 동일할 경우 백업할 } /home/oslab/P1/b.jpg /home/oslab/backup/P1/b.jpg 와 백업되어 있는 가 완전히 동일하지 않을 경우

case1의 결과

"/home/oslab/backup/P1/a.txt_230227172233" backedup

"/home/oslab/backup/P1/b.jpg_230227172231" is already backedup

"/home/oslab/backup/P1/D1/b.jpg_230227172233" backedup

case2의 결과

"/home/oslab/backup/P1/a.txt_230227172233" backedup

"/home/oslab/backup/P1/b.jpg_230227172233" backedup //백업 시간이 다르게 저장

"/home/oslab/backup/P1/D1/b.jpg_230227172233" backedup

20230000> add D1/a.txt help -

Usage : add <FILENAME> [OPTION]

-d : add directory recursive

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 () Usage 없을 경우 포함 경우 포함 입력 시 출력 후 프롬프트 재출력 점(2)
- 첫 번째 인자로 입력받은 경로() 절대 경로 가 길이 제한(4,096 Byte)을 넘거나 해당 경로가 일반 파일이나 디렉토리가 아니거나 접근권한이 없는 경우 에러 처리 후 프롬프트 재출력 점(2)
- 첫 번째 인자로 입력받은 경로() (절대 경로 가 사용자의 홈 디렉토리 \$HOME 또는 ~) 벗어나거나 백업 디렉토리 혹은 백업 디렉토리 내 파일 및 디렉토리를 포함한다면 입력 "< > 받은 경로 표 can't be backedup" (2 준 출력 후 프롬프트 재출력 점)
- 두 번째 인자로 올바르지 않은 옵션이 들어왔을 경우 출력 Usage (2) 후 프롬프트 재출력 점
- 첫 번째 인자가 디렉토리일 경우 옵션을 사용하지 않 '-d' (2) 있다면 에러 처리 후 프롬프트 재출력 점 - 백업파일의 기존 경로가 같고 해시, 값이 같은 파일이 이미 백업디렉토리에 존재한다면 백업을 하지 않고 백업파일의 "< 절대 경로 표 > is already backedup" (2) 준 출력 후 프롬프트 재출력 점

3. 내장명령어 2. remove

1) Usage : remove <FILENAME> [OPTION]

- 백업된 파일 을 입력 (FILENAME) (/home/backup 받아 해당 파일 또는 디렉토리를 백업디렉토리 디폴트 백업디렉토리는 임)에서 삭제

2) 인자 설명

- 첫 번째 인자 은 백업 디렉토리 내 백업된 파일이나 디렉토리의 상대경로와 <FILENAME> 절대경로 모두 입력 가능해야함 - 두 번째 인자 은 와 가 [OPTION] '-a' '-c' 있으며 모두 생략 가능함 와 옵션은 아래 설명 (-a -c) 확인 3) 실행결과
- 첫 번째 인자 에 대해 백업디렉토리에 <FILENAME> 있는 백업 파일 정(, a) 규파일만 디렉토리는 옵션 처리 - 중 인자로 주어 긴 파일의 경로가 과 일 <FILENAME> 치하는 백업파일을 삭제함 만약 백업파일이 여러개 존 . 재 시 "backup file list of "< > 원본 파일 경로 출력 "" () 후 다음 줄부터 리스트로 백업 파일의 백업 시간과 크기 천 단위로 쉼표 출력 를 출력하고 사용 자 입력 대기
- 첫 번째 인자로 상대경로를 입력했을 경우 입력한 상대경로에 대한 백업파일을 찾아야 하기 때문에 첫 번째 인자로 입력받 은 상대경로를 백업디렉토리에 맞게 변환해주는 과정 필요

- 사용자가 입력한 번호에 맞는 백업파일을 삭제 진행 후 결과 출력 및 프롬프트 재출력 번 입력시 명령어 실행 . 0 종료 후

- 5 -

프롬프트 재출력

그림 디렉토리 트리 구조의 예 3. /home/oslab/backup/	그림 디렉토리 트리 구조의 예 4. /home/oslab/
<pre> /home/oslab/backup/ └─ P1/ a/ └─ a.t └─ xt_20322172302 a.t └─ xt_20322172336 b/ └─ c.t └─ └─ xt_20322172302 b.t └─ xt_190315124532 b.t └─ xt_230116132511 └─ d.txt_20322182157 └─ d.txt_20322193011 └─ d.txt_2032210423 a.t └─ xt_203227172231 a.t └─ xt_203227172236 </pre>	<pre> /home/oslab/ └─ P1/ a/ └─ a.t └─ xt b/ └─ └─ c.t └─ └─ xt a.t └─ xt └─ backup/ </pre>

예 내 . remove (3 4 (pw 장명령어 실행 그림 과 의 상황에서 현재 작업 디렉토리 가 일 때 d) “/home/oslab/P1/”)
<pre> 20230000> remove Usage : remove <FILENAME> [OPTION] -a : remove all file(recursive) -c : clear backup directory 20230000> remove a.txt backup file list of "/home/oslab/P1/a.txt" // 2 동일한 파일이 개 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" // 2 동일한 파일이 개 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 </pre>

- ‘-a’ 옵션 입력 시에는 기존경로가 과 일 <FILENAME> 치하는 모든 백업파일을 삭제함 이 때 이 디렉토리라면 .
<FILENAME> 기존경로가 하위 파일인 백업 파일을 모두 삭제함 <FILENAME>

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

```
/home/oslab/backup/  
└─ P1/  
  a/ └─  
    a.t | └─ xt_20322172302  
    b.t | └─ xt_20322172302  
    b/ | └─  
    c.t | └─ └─ xt_20322172302  
    c.t | └─ xt_20322172302  
    a.t └─ xt_20322172231
```

예 내 . remove a (5 (pw 장명령어의 옵션 그 – 림 의 상황에서 현재 작업 디렉토리 가 일 때 d) “/home/oslab/P1/”)

```
20230000> remove a a – // remove /home/oslab/P1/a a ( ) – 와 동일한 결과 절대경로 입력  
“/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 출력 “backup directory clear” 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 디렉토리는 삭제되면 안됨 + . , /home/backup .

그림 디렉토리 트리 구조의 예 6. /home/oslab/backup/

```
/home/oslab/backup/  
└─ P1/  
  a/ └─  
    a.t | └─ xt_20322172302  
    b/ | └─  
    c.t | └─ xt_20322172302  
    a.t └─ xt_20322172231
```

예 내 . remove c (장명령어의 옵션 그 – 림 의 상 6 황에서 현재 작업 디렉토리 가 일 때 (pwd) “/home/oslab/P1/”)

```
20230000> remove a a c – –  
Usage : remove <FILENAME> [OPTION]  
-a : remove all file(recursive)  
-c : clear backup directory
```

```
20230000> remove c–  
backup directory cleared(3 regular files and 3 subdirectories totally). // 4 P1, P1/a/, 그림 의 디렉토리의 구조상 P1/a/b/ 3  
P1/a.t 등 개 디렉토리와 등 개의 정 xt_, P1/a/a/txt_, P1/a/b/c.txt_ 3 규파일이 백업되어 있을 경우
```

```
20230000> remove -c
```

no file(s) in the backup // , 백업 디렉토리에 아무 파일도 없을 경우 예외 처리 번째 7 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 () Usage 없을 경우 포함 입력 시 출력 후 프롬프트 재출력 점(3)
- 첫 번째 인자로 입력받은 경로() 절대 경로 가 길이 제한(4,096 Byte)을 넘거나 해당 경로가 일반 파일이나 디렉토리가 아니거나 접근권한이 없는 경우 예외 처리 후 프롬프트 재출력 점(5)
- 첫 번째 인자로 입력받은 경로() (절대 경로 가 사용자의 홈 디렉토리 \$HOME 또는 ~) 벗어나거나 백업 디렉토리 혹은 백업 디렉토리 내 파일 및 디렉토리를 포함한다면 입력 "< > 받은 경로 표 can't be backedup" (5 준 출력 후 프롬프트 재출력 점)
- 첫 번째 인자가 디렉토리인 경우 옵션을 사용하지 않 '-a' (3) 있다면 예외 처리 후 프롬프트 재출력 점 - 두 번째 인자로 올바르지 않은 옵션이 들어왔거나 옵션과 옵션이 '-a' '-c' Usage 동시에 들어왔을 경우 출력 후 프롬프트 재출력 점(3)
- '-c' Usage (2) 옵션 사용 시 다른 인자를 입력하면 출력 후 프롬프트 재출력 점
- '-c' 옵션 사용 시 백업디렉토리에 아무 파일이 없을 경우 출력 "no file(s) in the backup" (2) 후 프롬프트 재출력 점

- 7 -

4. 내장명령어 3. recover

1) Usage : recover <FILENAME> [OPTION]

- 백업된 파일 을 입력 (FILENAME) (/home/backup 받아 해당 파일 또는 디렉토리를 백업디렉토리 디폴트 백업디렉토리는 임)에서 파일 복구 진행

2) 인자 설명

- 첫 번째 인자 은 백업 디렉토리 내 백업된 파일이나 디렉토리의 상대경로와 <FILENAME> 절대경로 모두 입력 가능해야함 - 두 번째 인자 은 와 가 [OPTION] '-d' '-n' 있으며 모두 생략 가능함 와 옵션은 아래 설명 (-d -n) 확인 3)

실행결과

- 첫 번째 인자 에 대해 백업디렉토리에 <FILENAME> 있는 백업 파일 정(, 규파일만 디렉토리는 옵션 처리 -d) 중 인자로 주어 진 파일의 경로가 과 일 <FILENAME> 치하는 백업파일을 복구함 만약 백업파일이 여러개 존 . 재 시 "backup file list of "< > 원본 파일 경로 출력 "" () 후 다음 줄부터 리스트로 백업 파일의 백업 시간과 크기 천 단위로 쉼표 출력 를 출력하고 사용 자 입력 대기
- 첫 번째 인자로 상대경로를 입력했을 경우 입력한 상대경로에 대한 백업파일을 찾아야 하기 때문에 첫 번째 인자로 입력받 은 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 번 입력시 명령어 실행 .0 종료 후 프롬프트 재출력

그림 디렉토리 트리 구조의 예 7. /home/oslab/backup/	그림 디렉토리 트리 구조의 예 8. /home/oslab/
<pre> /home/oslab/backup/ ├── P1/ │ ├── a/ │ │ ├── xt_20322172302 │ │ └── b/ │ │ └── xt_20322172302 │ │ ├── d.txt_20322182157 │ │ ├── d.txt_20322193011 │ │ └── d.txt_2032210423 │ └── a.t_203227172231 │ └── a.t_203227172236 </pre>	<pre> /home/oslab/ ├── P1/ │ ├── a/ │ │ ├── xt │ │ └── b/ │ │ └── xt │ └── c.t_203227172231 │ └── a.t_203227172236 │ └── backup/ </pre>

예 내 . recover (장명령어 실행 그림 과 의 상 7 8 황에서 현재 작업 디렉토리 가 일 때 (pwd) "/home/oslab/P1/")

20230000> recover

Usage : recover <FILENAME> [OPTION]

-d : recover directory recursive

-n <NEWNAME> : recover file with new name

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 인 디렉토리의 모든 백업파일과 서브 디렉토리 내부의 파일들 <FILENAME> 까지 재귀적

- 8 -

으로 복구함

그림 디렉토리 트리 구조의 예 9. /home/oslab/backup/	그림 디렉토리 트리 구조의 예 10. /home/oslab/
/home/oslab/backup/ └ P1/ a/ └ a.t └ xt_20322172302 b.t └ xt_20322172302 b.t └ xt_20322172313 b/ └ c.t └ xt_20322172302 c.t └ xt_20322172302 a.t └ xt_20322172231	/home/oslab/ └ P1/ a/ └ a.t └ xt b/ └ c.t └ xt a.t └ xt └ backup/

예 내 .recover 장명령어의 옵션 그 -d (림 와 의 상 9 10 (pw 황에서 현재 작업 디렉토리 가 일 때 d) "/home/oslab/P1/")

```

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 내 디렉토리 내 에 대한 백업 파일이 개일 경우 9) P1/a b.txt
2 "/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 과 일 <FILENAME> <NEWFILE> 치하는 백업파일을 경로로 바꾸어 복구를 진행하고 의 파일 디렉토리가 <NEWFILE> / . 없다면 생성 후 덮어쓰 '-d' 입력과 동시에 사용할 시에는 기존경로가 <FILENAME> <NEWFILE> 인 디렉토리의 모든 백업파일을 경로로 복구를 진행 .

그림 디렉토리 트리 구조의 예 11. /home/oslab/backup/	그림 디렉토리 트리 구조의 예 12. /home/oslab/
<pre> /home/oslab/backup/ ├─ P1/ │ ├─ a/ │ │ ├─ a.t └─ xt_20322172302 │ │ └─ b.t └─ xt_20322172302 │ └─ b/ │ └─ c.t └─ xt_20322172302 │ └─ a.t └─ xt_20322172231 │ └─ a.t └─ xt_203227172236 </pre>	<pre> /home/oslab/ ├─ P1/ │ ├─ a/ │ │ ├─ a.t └─ xt │ │ └─ b/ └─ │ │ └─ c.t └─ xt │ └─ a.t └─ xt │ └─ backup/ </pre>

- 9 -

예 내 . recover n (11 12 (pw 장명령어의 옵션 그 - 림 과 의 상황에서 현재 작업 디렉토리 가 일 때 d) "/home/oslab/P1/")

```

20230000> recover a.txt n-
Usage : recover <FILENAME> [OPTION]
-d : recover directory recursive
-n <NEWNAME> : recover file with new name

```

```

20230000> recover a.txt n new_a.t - xt
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 내 . recover 장명령어의 옵션 그 -- d, n (11 12 (pw 림 과 의 상황에서 현재 작업 디렉토리 가 일 때 d) "/home/oslab/P1/")

```
20230000> recover a -- d n new_a
"/home/oslab/backup/P1/a/a.txt_230227172302" backup recover to "/home/oslab/P1/new_a/a.txt"
"/home/oslab/backup/P1/a/b.txt_230227172302" backup recover to "/home/oslab/P1/new_a/b.txt"
"/home/oslab/backup/P1/a/c.txt_230227172302" backup recover to "/home/oslab/P1/new_a/c.txt"
"/home/oslab/backup/P1/a/b/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 () Usage 없을 경우 포함 입력 시 출력 후 프롬프트 재출력 점(3)
- 첫 번째 인자로 입력받은 경로() 절대 경로 혹은 옵션 '-n' 뒤에 인자가 <NEWFILE> 길이 제한(4,096 Byte)을 넘거나 해당 경로가 일반 파일이나 디렉토리가 아니거나 접근권한이 없는 경우 에러 처리 후 프롬프트 재출력 점(5) - 첫 번째 인자로 입력받은 경로() 절대 경로 혹은 옵션 '-n' 뒤에 인자가 사용자의 <NEWFILE> (홈 디렉토리 %HOME 또는 ~)를 벗어나거나 백업 디렉토리 혹은 백업 디렉토리 내 파일 및 디렉토리를 포함한다면 입력 "< > 받은 경로 can't be backedup" (5) 표준 출력 후 프롬프트 재출력 점
- 첫 번째 인자가 디렉토리일 경우 옵션을 사용하지 않 '-d' (3) 있다면 에러 처리 후 프롬프트 재출력 점
- 두 번째 인자로 올바르지 않은 옵션이 들어왔을 경우 출력 Usage (3) 후 프롬프트 재출력 점
- 백업파일의 기존 경로가 과 <FILENAME> 같고 해당 파일과 해시 , (5) 값이 같다면 백업 복구를 진행하지 않음 점 - 기존 경로에 파일이 없다면 백업파일 상대로 생성 점(3)

5. 내장명령어 4. ls, vi(m)

1) Usage : ls, 기존 시스템 함수 과 vi(m) 동일

- exec() ls, 류 함수를 이용하여 시스템 함수 을 실행 함수 사용 vi(m) (system()) 불가

2) 인자 설명

- 기존 시스템 함수 과 ls, vi(m) 동일

3) 실행결과

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

예 실행 시 표 . list 준 출력

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

6. 내장명령어 5. help

1) Usage : help

- 프로그램 내장명령어에 대한 설명 출력 (Usage)

2) 실행결과

- 프로그램 내장명령어 실행 help

예 내 . help 장명령어 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 프로그램 ssu_backup 종료

2) 실행결과

- 프로그램 종료

예 실행 시 표 . exit 준 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 : , errno 오류가 발생 상세한 오류 내용은 에 설정

0 : , namelist struct 이상 정상적으로 처리 에 저장된 의 개수가 dirent * return

- 3. realpath : 상대경로를 절대경로로 변환하는 라이브러리 함수

```

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

NULL : , errno 오류가 발생 상세한 오류 내용은 전역변수에 설정
NULL이 아닌 경우 가 이 아 : resolved_path NULL 니면 를 , resolved_path return,

```

resolved_path가 이면 으로 할당하여 를 NULL , malloc(3) real path return 저장한 후에

- 11 -

- 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 execlv(const char *pathname, char *const argv[]);
int execlp(const char *pathname, const char *arg, .../* (char *) NULL, char *const envp[] */); int
execve(const char *pathname, char *const argv[], char *const envp[]);
int execlp(const char *file, const char *arg, .../* (char *) NULL */);
int execvp(const char *file, char *const argv[]);
int execvp(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 S 및 HA1

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

- 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 옵션 "-lcrypto" 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 S 해시값을 구하기 위해 를 사용 HA1(openssl/sha.h)

- 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 옵션 "-lcrypto" 필요
- SHA1() <https://www.openssl.org/> 사용법은 및 docs/man1.1.1/man3/SHA1.html <https://github.com/Chronic-Dev/openssl/blob/master/crypto/sha/sha1.c> 참고

o 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() : (shell) 명령 타겟을 만들기 위해 필요한 명령

Macro
Target : Dependency1 Dependency2 ...
<-Tab->Command 1
<-Tab->Command 2
<-Tab->...

- Makefile의 동작 순서

- 12 -

- ✓ make 사용 시 타겟을 지정하지 않으면 제일 처음의 타겟을 수행
- ✓ 타겟과 종속 항목들은 관습적으로 파일명을 명시
- ✓ 명령 항목들이 충족되었을 때 타겟을 생성하기 위해 명령 라인의 (command) 맨 위부터 순차적으로 수행
- ✓ 종속 항목의 마지막 수정 시간(st_mtime)을 비교 후 수행

- Makefile 작성 시 참고사항

- ✓ 명령의 시작은 반드시 으로 시작해야함 Tab
- ✓ 한 줄 주석은 여러 #, 줄 주석은 \
- ✓ 자동 매크로 현: 재 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

(1 예).Makefile	(2). 예 매크로를사용한경우	(3). 예 자동매크로를사용한경우
<pre>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</pre>	<pre>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</pre>	<pre>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 \$^</pre>

- (4). 예 수행 예시 Makefile

<pre>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:~\$</pre>	<pre>oslab@a-VirtualBox:~\$ make clean rm test.o rm add.o rm sub.o rm test oslab@a-VirtualBox:~\$</pre>
---	---

○ 보고서 제출 시 유의 사항

- 보고서 제출 마감은 제출일 11:59PM까지 구() 글 서버시간
- 지연 제출 시 감점 일 지 : 1 30 연 시 감점 일 이 % , 2 후 미제출 처리
- 압축 오류, s 파일 누락 관련 감점 yllabus 참고

- 필수구현 내 : 1. ssu_backup, 2. a 장명령어 예외처리는 dd, recover(. ,) 필수구현 아님 단 별도 감점 있음 - 배점 점
만점 실행 여부 배점 점으로 (100 . 80 , 15 5 ,) 최종 환산하며 보고서 점과 소스코드 주석 점은 별도 강의계획서 확인 1.
ssu_backup - 10점

- 13 -

2. 1. a 내장명령어 점 dd - 15

✓ '-d' 옵션

3. 2. remo 내장명령어 점 ve - 30

✓ '-a' 옵션

✓ '-c' 옵션

4. 3. reco 내장명령어 점 ver - 30

✓ '-d' 옵션

✓ '-n' 옵션

- 내장명령어 점 4. ls, vi(m) - 5

- 내장명령어 점 5. help - 3

- 내장명령어 점 6. exit - 2

- makefile () 5 작성 매크로 사용하지 않아도 됨 점

-

