설계과제 3 개요 : Soongsil Backup

Linux System Programming, School of CSE, Soongsil University, Spring 2019

□ 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 자료구조를 이용하여 프로그램을 작성함으로써 시스템 프로그래밍 설계 및 응용 능력을 향상

□ 팀 구성
- 개인별 프로젝트

□ 보고서 제출 방법
- 설계과제는 ” 보고서.hwp“ (개요, 상세설계, 구현방법, 결과 및 소스코드와 실행결과가 함께 있는 워드(hwp 또는 MS-Word) 파일))와 ” 소스코드“ (makefile, obj, *.c, *.h 등 컴파일하고 실행하기 위한 모든 파일)를 제출해야 함
- 모든 설계과제 결과물은 “ #P설계과제번호_학번_버전.zip”(예. #P3_20190000_v1.0.zip)형태로 파일 이름을 명명하고, zip프로그램으로 압축하여 제출해야 함.
- 압축파일 내 “ 보고서” 디렉토리와 “ 소스코드” 디렉토리 2개 만들어 제출해야 함
- 제출한 압축 파일을 풀었을 때 해당 디렉토리에서 컴파일 및 실행이 되어야 함. 해당 디렉토리에서 컴파일이나 실행되지 않을 경우, 기본과제 및 설계과제 제출 방법을 따르지 않는 경우 감점 20% 외 추가 20% 감점
- 기타 내용은 syllabus 참고

□ 제출 기한
- 6월 3일(월) 오후 11시 59분 59초 (서버 시간이 30분 정도 빠를 수 있기 때문에 1시간 지연 허용)

□ 보고서 양식
- 보고서는 다음과 같은 양식으로 작성

1. 과제 개요 // 위 개요를 더 상세하게 작성
2. 설계 // 함수 기능별 흐름도(순서도) 반드시 포함
3. 구현 // 함수 프로토타입 반드시 포함
4. 테스트 및 결과 // 테스트 프로그램의 실행 결과 캡쳐 및 분석
5. 소스코드 // 주석

□ ssu_backup 프로그램 기본 사항
- 하나의 쓰레드가 한개 파일의 백업을 담당
- ssu_backup 프로그램을 실행 할 때 지정한 경로에 백업 디렉토리를 생성하고, 사용자가 지정한 백업 주기마다 백업해야할 파일을 백업 디렉토리에 백업
- 정상적인 수행 로그는 별도로 생성한 하나의 로그파일에 기록
  - ✓ 생성된 백업 디렉토리에 하나의 로그파일 생성 및 기록
  - ✓ [수행시간] 수행내용 형태로 작성
  - ✓ add, remove, recover 명령어 실행 성공 시 로그파일에 기록
  - ✓ 백업파일 생성 성공 시 로그파일에 기록
  - ✓ 시간 순서대로 로그가 작성될 수 있도록 동기화 수행

예. 로그 파일 작성 예시

```
[190311 153320] /home/kyg/lsp_p3/test1.txt added
[190311 153325] /home/kyg/lsp_p3/test1.txt_190311153325 generated
[190311 153330] /home/kyg/lsp_p3/test1.txt_190311153330 generated
[190311 153333] /home/kyg/lsp_p3/test2.txt added
[190311 153335] /home/kyg/lsp_p3/test1.txt_190311153335 generated
[190311 153340] /home/kyg/lsp_p3/test1.txt_190311153340 generated
[190311 153343] /home/kyg/lsp_p3/test2.txt_190311153343 generated
[190311 153345] /home/kyg/lsp_p3/test1.txt_190311153345 generated
[190311 153349] /home/kyg/lsp_p3/test1.txt deleted
[190311 153353] /home/kyg/lsp_p3/test2.txt_190311153353 generated
[190311 153403] /home/kyg/lsp_p3/test2.txt_190311153403 generated
[190311 153410] /home/kyg/lsp_p3/test2.txt deleted
```

- 백업해야할 파일의 이름은 기존 백업 파일의 절대경로 바로 뒤에 이어 “_백업시간”(YYMMDDHHMMSS)을 붙임
  - ✓ 연도(YY)는 4자리 중 뒤 2자리만 사용(2019 중 19만 사용)
- 백업해야할 파일 절대경로에는 한글을 포함시키면 안됨(ASCII로 표현할 수 있는 알파벳을 절대경로로 사용)
- 백업해야할 파일 이름(절대경로)이 길이 제한(255 byte)을 넘을 시 에러 처리 후 프롬프트로 제어가 넘어감

□ 설계 및 구현

- ssu_backup
  - ✓ ssu_backup 실행 시 백업 디렉토리를 생성할 경로를 인자로 입력
    - 상대경로와 절대경로 모두 입력 가능
    - 인자가 없으면 current working 디렉토리 밑에 백업 디렉토리 생성· 백업
    - 인자가 2개 이상이면 usage 출력 후 프로그램 종료
    - 인자로 입력받은 디렉토리를 찾을 수 없으면 usage 출력 후 프로그램 종료
    - 인자로 입력받은 디렉토리가 디렉토리 파일이 아니라면 usage 출력 후 프로그램 종료
    - 인자로 입력받은 디렉토리의 접근권한이 없는 경우 usage 출력 후 프로그램 종료
  - ✓ ssu_backup 실행 후 다음과 같은 프롬프트 출력
    - 프롬프트 모양 : 공백없이 학번, ‘ >’ 문자 출력. ex) 20190000>
    - 프롬프트에서 실행 가능 명령어 : add, remove, compare, recover, list, ls, vi(m), exit
    - 이외 명령어 입력 시 에러 처리 후 프롬프트로 제어가 넘어감
    - 엔터만 입력 시 프롬프트 재출력
    - exit 명령이 입력될 때까지 위에서 지정한 실행 가능 명령어를 입력받아 실행

- add <FILENAME> [PERIOD] [OPTION]
  - ✓ 백업해야할 파일(FILENAME)을 백업 리스트에 새롭게 추가
  - ✓ 백업 리스트는 링크드 리스트로 구현
  - ✓ 백업 리스트 구조체는 1. 파일의 절대경로, 2. 백업 주기, 3. 백업 옵션 등을 포함
  - ✓ FILENAME
    - 백업해야할 파일의 경로(절대경로와 상대경로 모두 입력 가능해야 함)
    - FILENAME 입력 없을 시 에러 처리 후 프롬프트로 제어가 넘어감
    - 백업해야할 파일이 존재하지 않을 시 에러 처리 후 프롬프트로 제어가 넘어감
    - 백업해야할 파일은 일반 파일만 가능하며, 일반 파일이 아니라면 에러 처리 후 프롬프트로 제어가 넘어감
    - 백업해야할 파일이 이미 백업 리스트에 존재한다면 에러 처리 후 프롬프트로 제어가 넘어감
  - ✓ PERIOD

- 백업해야할 파일에 대한 백업주기
- PERIOD 입력 없을 시 에러 처리 후 프롬프트로 제어가 넘어감
- PERIOD는 정수형(채점의 편의를 위해 제출할 예제 프로그램에서는 5≤PERIOD≤10로 설정하여 실행시켜 제출), 정수형이 아닌 실수형 입력 시 에러 처리 후 프롬프트로 제어가 넘어감

✓ 옵션 [OPTION]

- 옵션이 없는 경우 : 백업해야할 파일을 주어진 PERIOD마다 백업 실행

실행 예. 옵션이 없는 경우

```
20190000>add test1.txt 5
20190000>list
/home/kyg/lsp_p3/test1.txt
```

- 아래 -m, -n, -t, -d 옵션 모두 동시에 사용 가능해야 함
- 여러 개 옵션을 동시에 사용 시 옵션마다 ' -' 이 붙어야 함
- -m : 입력받은 PERIOD마다 파일의 mtime이 수정 되었을 경우 백업 실행

- -n NUMBER : NUMBER는 백업한 파일의 최대 개수. 가장 최근 NUMBER개의 백업파일 외 나머지 파일은 삭제
  (1) NUMBER는 정수형이며, 정수형이 아닌 실수형 입력 시 에러 처리 후 프롬프트로 제어가 넘어감
  (2) NUMBER 입력 없을 시 에러 처리 후 프롬프트로 제어가 넘어감
  (3) 채점의 편의를 위해 제출할 예제 프로그램에서는 1≤NUMBER≤100로 설정하여 실행시켜 제출

- -t TIME : 백업해야할 파일에 대한 백업 디렉토리 내 보관 기간을 TIME만큼 설정
  (1) TIME는 정수형이며 초를 나타냄. 정수형이 아닌 실수형 입력 시 에러 처리 후 프롬프트로 제어가 넘어감
  (2) TIME 입력 없을 시 에러 처리 후 프롬프트로 제어가 넘어감
  (3) 매 PERIOD마다 백업 파일을 생성하고 기존 모든 백업 파일의 생성시간을 확인. 확인한 파일의 생성시간이 주어진 TIME보다 크면 해당 파일 삭제
  (4) 채점의 편의를 위해 제출할 예제 프로그램에서는 60≤TIME≤1200로 설정하여 실행시켜 제출

- -d : 지정한 디렉토리 내의 모든 파일들을 백업 리스트에 추가
  (1) 한번에 인자로 줄 수 있는 디렉토리는 최대 1개이며 해당 디렉토리 내 모든 파일을 리스트에 추가. 디렉토리 내의 서브디렉토리 내 모든 파일까지 모두 리스트에 추가
  (2) FILENAME이 디렉토리가 아닐 경우 에러 처리 후 프롬프트로 제어가 넘어감
  (3) 디렉토리 내에 있는 파일이 이미 백업 리스트에 존재할 경우 해당 파일은 리스트에 추가하지 않고 건너뜀

실행 예. -d 옵션

```
20190000>add -d testdir 5
20190000>list
/home/kyg/lsp_p3/testdir/test1.txt      5
/home/kyg/lsp_p3/testdir/test2.txt      5
/home/kyg/lsp_p3/testdir/test3.txt      5
```

- remove <FILENAME> [OPTION]

✓ 백업 리스트에 존재하는 파일(FILENAME)의 백업을 중단하기 위해 백업 리스트에서 삭제

✓ FILENAME

- 백업을 하지 않을(중단할) 파일의 경로(절대경로와 상대경로 모두 입력 가능해야 함)
- FILENAME 입력 없을 시 에러 처리 후 프롬프트로 제어가 넘어감
- 백업을 중단할 파일이 백업 리스트에 존재하지 않을 시 에러 처리 후 프롬프트로 제어가 넘어감

✓ 옵션 [OPTION]
- 옵션이 없는 경우 : 백업을 하지 않을(중단할) 파일을 백업 리스트에서 제거
- -a : 현재 백업 실행중인 모든 파일을 백업 리스트에서 제거

(1) -a 옵션 입력 시 FILENAME은 입력하지 않음

(2) FILENAME 입력 시 에러 처리 후 프롬프트로 제어가 넘어감

실행 예. remove 명령어의 -a 옵션

```
20190000>add -d testdir 5
20190000>list
/home/kyg/lsp_p3/testdir/test1.txt      5
/home/kyg/lsp_p3/testdir/test2.txt      5
/home/kyg/lsp_p3/testdir/test3.txt      5
20190000>remove -a
20190000>list
20190000>exit
```

- compare <FILENAME1> <FILENAME2>

✓ FILENAME1과 FILENAME2의 mtime과 파일 크기 비교. 백업한 파일, 백업할 파일 모두 적용 가능

✓ FILENAME이 존재하지 않거나 일반 파일이 아닐 경우 에러 처리 후 프롬프트로 제어가 넘어감

✓ 입력 인자가 2개가 아닐 경우 에러 처리 후 프롬프트로 제어가 넘어감

✓ mtime과 파일 크기가 같은 경우 두 파일은 동일한 파일로 취급

✓ 두 파일이 동일할 경우 표준출력으로 동일하다는 문구 출력

✓ 두 파일이 동일하지 않을 때 표준출력으로 각 파일의 mtime과 파일 크기 출력

- recover <FILENAME> [OPTION]

✓ FILENAME의 백업 파일을 사용하여 현재의 파일(FILENAME)을 백업된 파일로 변경

✓ 백업파일이 존재하는 경우 백업파일의 백업시간(YYMMDDHHMMSS), 파일크기를 리스트 형태로 백업시간 기준 오름차순 출력

✓ 리스트는 사용자가 선택할 수 있도록 순번이 있으며, 파일을 선택하지 않는 경우(순번)도 생성

✓ 사용자가 백업파일을 선택하면 변경된 파일의 내용을 출력

✓ FILENAME
- 변경할 파일의 경로(절대경로와 상대경로 모두 입력 가능해야 함)
- 변경할 파일이 존재하지 않으면 에러 처리 후 프롬프트로 제어가 넘어감
- 변경할 파일이 현재 백업 리스트에 존재한다면 백업 수행 종료 후 복구 진행
- 변경할 파일에 대한 백업 파일이 존재하지 않으면 에러 처리 후 프롬프트로 제어가 넘어감

✓ 옵션 [OPTION]
- -n <NEWFILE> : 백업파일을 원본파일로 덮어쓰지(변경하지) 않고 새로운 파일을 생성

(1) NEWFILE은 경로를 포함한 새로운 파일의 이름(상대경로와 절대경로 모두 입력 가능해야 함)

(2) NEWFILE 입력 없을 시 에러 처리 후 프롬프트로 제어가 넘어감

(3) NEWFILE이 이미 존재한다면 에러 처리 후 프롬프트로 제어가 넘어감

실행 예. recover 명령어의 -n 옵션

```
20190000>recover ./testdir/test2.txt -n ./testdir/test2_recover.txt
0. exit
1. 190423231011         13bytes
2. 190423231016         13bytes
3. 190423231021         17bytes
4. 190423231026         10bytes
5. 190423231031         16bytes
Choose file to recover : 4
Recovery success
20190000>ls testdir
test1.txt  test2.txt  test2_recover.txt  test3.txt
20190000>
```

- list
  - ✓ 현재 백업 실행중인 모든 백업 리스트 출력
  - ✓ 한 줄에 한 개 파일의 절대경로, PERIOD, OPTION 출력

- ls, vi(m)
  - ✓ system() 함수로 각각 " ls", " vi(m)" 명령어를 사용하여 구현

- exit
  - ✓ 현재 실행중인 모든 백업을 중지하고 프로그램 종료

<참고> 과제 구현에 필요한 함수(쓰레드 관련 함수 외 나머지 함수는 필수 사용 아님)

1. pthread_create() : p.354

```
#include <pthread.h>
int pthread_create(pthread_t *restrict tidp, const pthread_attr_t *restrict attr,
                   void *(*start_rtn)(void *), void *restrict arg);
                                                    리턴값: 성공 시 0, 실패 시 오류 번호
```

2. pthread_exit() : p.362

```
#include <pthread.h>
void pthread_exit(void *rval_ptr);
```

3. pthread_join(), pthread_detach() : p.365

```
#include <pthread.h>
int pthread_join(pthread_t thread, void **rval_ptr);
int pthread_detach(pthread_t tid);
                                                    리턴값: 성공 시 0, 실패 시 오류 번호
```

4. pthread_mutex_init(), pthread_mutex_destroy() : p.374

```
#include <pthread.h>
int pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexatt_t *attr);
int pthread_mutex_destroy(pthread_mutex_t *mutex);
                                                    리턴값: 성공 시 0, 실패 시 오류 번호
```

5. pthread_cond_init(), pthread_cond_destroy() : p.378

```
#include <pthread.h>
int pthread_cond_init(pthread_cond_t *restrict condition, pthread_condattr_t *restrict attr);
int pthread_cond_destroy(pthread_cond_t *condition);
                                                    리턴값: 성공 시 0, 실패 시 오류 번호
```

6. 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
```

7. system() : p.270

```
#include <stdlib.h>
int system(const char* string);
                                                    리턴값: 성공시 0이 아닌 값, 실패시 0
```

□ 보고서 제출 시 유의 사항
- 보고서 제출 마감은 제출일 자정까지 (1시간 지연 허용)
- 지연 제출 시 감점 : 1일 지연 시 마다 30% 감점, 3일 지연 후부터는 미제출 처리
- 압축 오류, 파일 누락 관련 감점 syllabus 참고

□ 구현 점수
가. ssu_backup 프롬프트 동작 10
나. 백업에 대한 별도의 로그파일 생성 및 기록 10
다. 로그파일 내용의 시간 순서 동기화 7
라. add 명령어 10
마. add -m 옵션 4
바. add -n 옵션 4
사. add -t 옵션 5
아. add -d 옵션 5
자. remove 명령어 10
차. remove -a 옵션 3
카. compare 명령어 8
타. recover 명령어 8
파. recover -n 옵션 3
하. list 명령어 10
거. exit 명령어 1
너. makefile 작성 2

□ **필수 구현 사항 : 가, 나, 라, 자, 하, 거**