설계과제 2 개요 : Soongsil monitoring

Linux System Programming, School of CSE, Soongsil University, Spring 2020

○ 개요

- ssu_mntr은 지정한 디렉토리를 모니터링하며 변경사항을 로그로 남기는 프로그램임

○ 목표

- 유닉스/리눅스 컴퓨팅 환경에서 프로세스가 동작하는 과정을 이해하고 파일속성 디렉토리에 관한 정보를 비교하여 자료구조를 이용해 프로그램을 작성하고 시스템 프로그램 설계 및 응용 능력을 향상시킴

○ 팀 구성

- 개인별 프로젝트

○ 보고서 제출 방법

- 설계과제는 ” 보고서.hwp “(개요, 분석, 상세설계, 구현방법, 결과 및 소스코드와 실행결과가 함께 있는 워드(hwp 또는 MS-Word) 파일))와 ” 소스코드 “(Makefile(필수), obj, *.c, *.h 등 컴파일하고 실행하기 위한 모든 파일을 제출해야 함
- 모든 설계과제 결과물은 “#P설계과제번호_학번_버전.zip” (예. #P2_20180000_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

- 5월 13일(수) 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_mntr 프로그램 기본 사항

- ssu_mntr 프로그램은 지정한 특정 디렉토리의 파일의 변경 상태 모니터링
  - ✓ 지정한 디렉토리는 제출할 소스코드 디렉토리 밑에 서브디렉토리로 생성
  - ✓ 모니터링 프로그램은 별도 생성한 디몬 프로세스가 담당
  - ✓ 파일이 생성, 삭제, 수정될 경우, 학번 서브디렉토리 밑에 있는 "log.txt" 파일에 변경 사항을 추가

예. log.txt 예시

```
oslab@mini:~/lsp/test/20162495$ cat log.txt
[2020-04-04 10:04:41][modify_test.txt]
[2020-04-04 10:05:05][modify_test.txt]
[2020-04-04 10:30:56][create_test.c]
[2020-04-04 10:51:23][delete_test.txt]
[2020-04-04 11:30:03][create_test.txt]
[2020-04-04 11:45:15][create_dir1_1.txt]
[2020-04-04 11:45:16][modify_dir1]
[2020-04-04 11:59:25][create_1.c]
[2020-04-04 12:14:34][modify_1.c]
[2020-04-04 12:15:59][modify_1.c]
[2020-04-04 12:21:13][create_2.c]
[2020-04-04 12:28:38][modify_dir1_1.txt]
[2020-04-04 12:28:40][modify_dir1]
[2020-04-04 12:40:11][create_3.c]
[2020-04-04 12:41:20][modify_3.c]
[2020-04-04 12:43:01][delete_3.c]
```

  - ✓ log.txt 파일 구조
    - [생성 or 삭제 or 수정 시간] [수행내용_파일이름] 형태로 작성
    - 새로운 파일 변경 사항은 "log.txt" 파일 밑에 추가
    - 파일 이름에는 한글 지원하지 않아도 됨
  - ✓ 모니터링이 종료될 경우, 종료 메시지(학생이 원하는 종료 메시지 마음대로)를 출력한 후 종료.

○ 설계 및 구현

- 가) ssu_mntr 실행 후 프롬프트 출력
  - ✓ 프롬프트 모양 : "학번>" 문자 출력. ex) 20201234>
  - ✓ 프롬프트에서는 delete, size, recover, tree, exit, help 명령어만 수행
  - ✓ 이외의 명령어 수행 시 자동으로 help를 실행시킨 것과 동일하게 표준 출력 후 프롬프트 출력
  - ✓ 엔터만 입력 시 프롬프트 재출력
- 나) DELETE [FILENAME] [END_TIME] [OPTION]
  - ✓ 지정한 삭제 시간에 자동으로 파일을 삭제해주는 명령어
  - ✓ 삭제한 파일은 제출할 소스코드 디렉토리 밑에 있는 'trash' 디렉토리로 이동
  - ✓ "trash" 디렉토리가 없을 경우 생성해야 함.
  - ✓ trash 디렉토리
    - trash 디렉토리 밑에 두 개의 서브디렉토리, files, info 디렉토리를 생성
      - "files" 디렉토리 : DELETE 명령어로 지운 파일 자체 저장 (rename으로 경로 변경)
      - "info" 디렉토리 : DELETE 명령어로 지운 파일의 정보(절대경로, 삭제 시간, 최종 수정시간(mtime))를 저장. 단 저장될 파일 이름은 절대경로를 제외하고 최종 파일 이름만 포함됨.

- info 디렉토리의 크기가 정해진 크기(2KB)를 초과할 경우 오래된 파일부터 files 디렉토리의 원본 파일과 함께 info 디렉토리 파일 정보까지 삭제

예. trash 디렉토리 예시, info안 파일 내용 예시

```
oslab@mini:~/lsp/test/20162495/trash$ ls
files  info
oslab@mini:~/lsp/test/20162495/trash$ cd info
oslab@mini:~/lsp/test/20162495/trash/info$ ls
3.txt
oslab@mini:~/lsp/test/20162495/trash/info$ cat 3.txt
[Trash info]
/home/oslab/lsp/test/20162495/check/3.txt
D : 2020-04-04 18:50:44
M : 2020-04-04 17:24:14
```

✓ FILENAME
- 파일의 경로(절대경로와 상대경로 모두 가능해야함)
- FILENAME 입력이 없거나 존재하지 않을 경우, 에러처리 후 프롬프트로 출력
- 삭제할 파일은 지정한 특정 디렉토리 안에 있는 파일들로 한정함

✓ END_TIME
- 삭제할 파일의 삭제 예정 시간 ex) 20201234> DELETE FILELNAME 2020-05-05 10:00 (5월 5일 10시 삭제)
- 단, 제출 실행 예제는 테스트가 쉽게 가능하면 3분 이내 시간을 사용하기 바람.
- END_TIME을 주어주지 않을 경우 바로 삭제함

✓ OPTION
- -i : 삭제 시 'trash' 디렉토리로 삭제 파일과 정보를 이동시키지 않고 파일 삭제
- -r : 지정한 시간에 삭제 시 삭제 여부 재확인

예. DELETE -r 옵션 예시

```
oslab@mini:~/lsp/test/20162495$ ./ssu_mntr
20201234>delete 1.c 2020-04-07 16:00 -r
Detlete [y/n]? n
```

- 다) SIZE [FILENAME] [OPTION]

✓ 파일경로(상대경로), 파일 크기 출력하는 명령어

✓ 기본 크기는 byte단위임

✓ 출력하는 파일 및 디렉토리는 문자열 오름차순으로 출력해야함

✓ FILENAME
- 디렉토리와 파일이 모두 가능하며 FILENAME으로 입력 시 크기가 모두 표시되어야함
- 파일경로는 상대경로로 출력해야함

예. SIZE의 출력 결과

```
oslab@mini:~/lsp/test/20162495$ ./ssu_mntr
20201234>size check
828     ./check
20201234>
```

✓ OPTION
- -d NUMBER : NUMBER 단계만큼의 하위 디렉토리까지 출력
  - NUMBER의 수보다 하위 디렉토리의 수가 적을 경우, 최대 하위 디렉토리까지 출력

예. SIZE의 -d number 출력 결과

```
oslab@mini:~/lsp/test/20162495$ ./ssu_mntr
20201234>size check -d 2
136     ./check/1.c
88      ./check/2.c
124     ./check/3.c
222     ./check/dir1
108     ./check/test.c
150     ./check/test.txt
20201234>
```

- 라) RECOVER [FILENAME] [OPTION]
  - ✓ “trash” 디렉토리 안에 있는 파일을 원래 경로로 복구하는 명령어
  - ✓ 동일한 이름의 파일이 “trash” 디렉토리에 여러 개 있을 경우 이를 표준 출력으로 파일 이름, 삭제 시간, 수정 시간을 보여줌
  - ✓ 복구 시 이름이 중복된다면 파일의 처음에 “숫자_” 를 추가

예. RECOVER의 이름 중복

```
oslab@mini:~/lsp/test/20162495$ ./ssu_mntr
20201234>recover 1.c
1. 1.c  D : 2020-04-04 18:09:23 M : 2020-04-04 17:24:14
2. 1.c  D : 2020-04-06 21:34:09 M : 2020-04-04 21:33:49
Choose : 2
```

  - ✓ FILENAME
    - 해당 파일이 없거나 파일의 복구 경로가 없다면 에러처리 후 프롬프트 전환

예. RECOVER의 에러 출력

```
oslab@mini:~/lsp/test/20162495$ ./ssu_mntr
20201234>recover dir1.c
There is no 'dir1.c' in the 'trash' directory
20201234>
```

  - ✓ OPTION
    - -l : ‘trash’ 디렉토리 밑에 있는 파일과 삭제 시간들을 삭제 시간이 오래된 순으로 출력 후, 명령어 진행

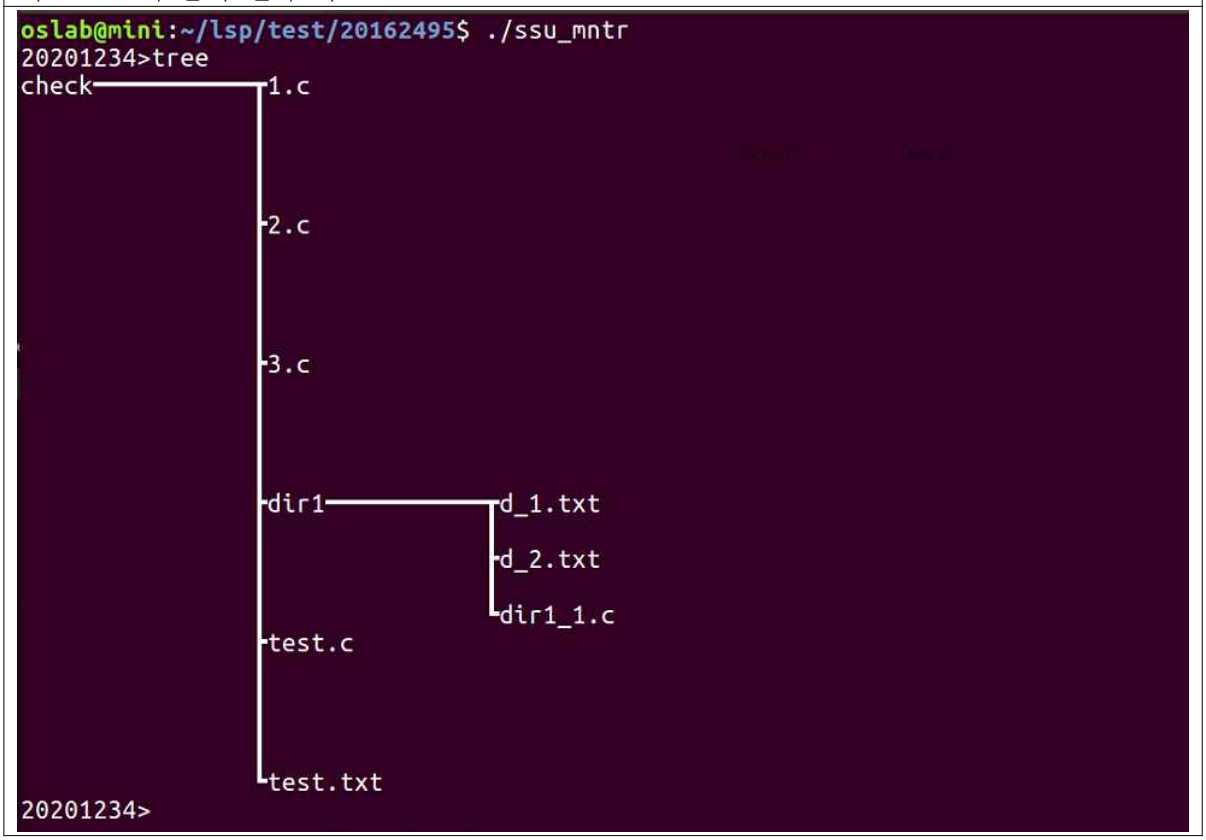
예. RECOVER의 i 옵션 출력 결과

```
oslab@mini:~/lsp/test/20162495$ ./ssu_mntr
20201234>recover 1.c -l
1. 1.c          2020-04-04 17:24:14
2. test.txt             2020-04-04 17:30:57
3. 3.c          2020-04-04 17:32:25
4. 2.c          2020-04-04 17:33:11
5. test.c               2020-04-04 17:34:19
```

- 마) TREE
  - ✓ “check” 디렉토리의 구조를 tree 형태로 보여주는 명령어
  - ✓ 꼭 이 형태를 따르지 않아도 되나 지정한 디렉토리의 구조를 그림으로 보여주어야 함
  - ✓ 아래 예시에서는 ~lsp/test/20162495/check 디렉토리가 파일의 변경 상태를 모니터링을 하기 위해 지정한 특정 디렉토리임

예. TREE의 출력 결과 리스트

-



- 바) EXIT
  - ✓ 프로그램 종료시키는 명령어
  - ✓ EXIT 명령어 실행 시 명령어를 실행시키는 프로세스와 지정한 디렉토리를 모니터링하는 프로세스 모두 종료되어야함.
- 사) HELP
  - ✓ 명령어 사용법을 출력하는 명령어

○ 과제 구현에 필요한 함수 (필수 아님)

- 1. 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
```

- 2. realpath : 상대경로를 절대경로로 변환하는 라이브러리 함수

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

                                                    NULL이 아닌 경우
                                        resolved_path가 NULL이 아니면, resolved_path를 return,
                        resolved_path가 NULL이면, malloc(3)으로 할당하여 real path를 저장한 후에 return
                                                    NULL
                                        오류가 발생, 상세한 오류 내용은 errno 전역변수에 설정
```

○ 보고서 제출 시 유의 사항

- 보고서 제출 마감은 제출일 자정까지 (30분 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_mntr 동작 - 20

나) delete 명령어 - 10

다) delete -i 옵션 - 5

라) delete -r 옵션 - 5

마) size 명령어 - 10

바) size -d 옵션 - 5

사) recover 명령어 - 20

아) recover -l 옵션 - 5

자) tree 명령어 - 10

차) exit 명령어 - 5

카) help 명령어 - 5

○ 필수 구현 사항 : 가, 나, 사, 자, 차