운영체제및실습 01분반
실습 7회차

정규표현식 & 동시성 문제

2021.05.05(수)

Contents

리눅스 명령어(6)

1. grep (regular expression, 정규표현식)

리눅스 명령어(6)

bash Expansion/Quoting 복습 (연습문제)

1. arithmetic expansion을 활용하여 echo로 3*(5+7)의 결과를 출력하는 명령줄은?

2. echo {a..b}_{c..d} {e..f}의 결과는?

3. 현재 디렉토리에 첫 문자가 대문자가 아닌 파일의 이름들을 모두 출력하는 echo 명령어는? (숨김 파일 제외)

4. 3번의 결과를 이용하여, 각 파일을 롱 리스팅 포맷으로 출력하고 싶다. 이 때 파일이 디렉토리라면 내부 목록은 들여다보고 싶지 않고, 디렉토리 그 자체를 보고 싶다. 명령어는?

5. 현재 디렉토리에 .txt로 끝나는 파일이 a.txt 하나 있고, username은 joonhee다.

아래 명령어의 각각의 결과는?

```
echo text ~/*.txt {a, b} $(echo foo) $((3*5*7)) $USER
echo “text ~/*.txt {a, b} $(echo foo) $((3*5*7)) $USER”
echo ‘text ~/*.txt {a, b} $(echo foo) $((3*5*7)) $USER’
```

리눅스 명령어(6)

- **bash Expansion/Quoting 복습 (연습문제 - 답안)**

```
09:31 $ echo $((3*(5+7)))
36
```

```
joonhee@joonhee-laptop:~$ echo {a..b}_{c..d} {e..f}
a_c a_d b_c b_d e f
```

```
joonhee@joonhee-laptop:~$ echo [![:upper:]]*
```

```
joonhee@joonhee-laptop:~$ ls -ld $(echo [![:upper:]]*)
drwxrwxr-x  6 joonhee joonhee   4096  4월 18 08:38  argos3
drwxrwxr-x  9 joonhee joonhee   4096  4월 18 08:37  argos3-examples
drwxrwxr-x  7 joonhee joonhee   4096  3월 26 12:20  catkin_ws
-rw-rw-r--  1 joonhee joonhee  24345  4월 14 10:44  dirlist-usr-bin.txt
```

```
joonhee@joonhee-laptop:~$ echo text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER
text /home/joonhee/two words.txt a b foo 4 joonhee
joonhee@joonhee-laptop:~$ echo "text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER"
text ~/*.txt {a,b} foo 4 joonhee
joonhee@joonhee-laptop:~$ echo 'text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER'
text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER
```

grep (regular expression, 정규표현식)

grep

: grep [options] ***regex*** [files...]

- 파일에서 regex(regular expression, 정규표현식)에 일치하는 줄들을 출력

options

-i (--ignore-case) : 대소문자 구분없이
-v (--invert-match) : 불일치 라인
-c (--count) : 일치하는 라인 수
-l (--files-with-matches) : 파일 이름
-L (--files without-match) : 불일치 파일이름
-n (--line-number) : 라인 번호 같이
-h (--no-filename) : 파일 이름 없이 (여러 파일일 때)

```
07:04 $ ls /usr/bin | grep zip
bunzip2
bzip2
bzip2recover
```

```
07:04 $ ls /usr/bin | grep Zip
```

```
07:06 $ ls /usr/bin | grep -i Zip
bunzip2
bzip2
bzip2recover
```

```
07:06 $ ls /usr/bin | grep -c zip
20
07:07 $ ls /usr/bin | grep -vc zip
2349
07:07 $ ls /usr/bin | wc -l
2369
```

```
07:07 $ ls /usr/bin > dirlist-usr-bin.txt
07:09 $ ls /usr/sbin > dirlist-usr-sbin.txt
07:10 $ grep -n bzip dirlist*.txt
dirlist-usr-bin.txt:158:bzip2
dirlist-usr-bin.txt:159:bzip2recover
07:09 $ grep -h bzip dirlist*.txt
bzip2
bzip2recover
```

```
07:13 $ ls -l /usr/bin | grep -l zip
(standard input)
```

```
07:12 $ grep -l bzip dirlist*.txt
dirlist-usr-bin.txt
07:12 $ grep -L bzip dirlist*.txt
dirlist-usr-sbin.txt
```

grep (regular expression, 정규표현식)

■ 정규표현식 (regular expression)

: 문자열을 나타내는 패턴

- 대표적으로 두 종류의 정규표현식 존재

  POSIX(BRE=basic, ERE=extended) / perl(PCRE)

일반적으로 싱글 쿼팅이 expansion을 완전 차단하므로 더 추천!

◆ Metacharacters (^ $. [ ] { } - ? * + ( ) | ₩)

shell에 인자로 넘길 때 expansion 고려해서 적절히 quoting해야 함

- . : the any character (아무 문자 하나, 길이 차지)
- ^ $: Anchors (^ : 라인의 시작, $: 끝)

^는 상관 없지만, $는 쿼팅해야겠죠??

```
07:29 $ grep -c zip dirlist-usr-bin.txt
20
07:30 $ grep -c .zip dirlist-usr-bin.txt
13
```

```
zip
zipcloak
zipdetails
zipgrep
zipinfo
zipnote
zipsplit
```

```
07:35 $ grep "zip$" dirlist-usr-bin.txt
funzip
gpg-zip
gunzip
```

```
07:36 $ grep "^zip" dirlist-usr-bin.txt
zip
zipcloak
zipdetails
```

```
07:42 $ ls -l /usr/bin | grep "zip$"
-rwxr-xr-x 1 root root       26776  8월 16  2019 funzip
```

```
07:36 $ grep "^zip$" dirlist-usr-bin.txt
zip
```

```
07:42 $ ls -l /usr/bin | grep "^zip"
```

출력없음!

```
07:43 $ grep -i '^..j.r$' /usr/share/dict/words
Major
major
```

grep (regular expression, 정규표현식)

- **grep - 연습문제**
  - **/usr/share/dict/words에서 2번째가 j, 4번째 글자가 k인 줄을 출력하시오.**
  - **/usr/bin에서 파일 이름에 gcc가 포함된 파일명을 출력하시오.**
  - **/usr/bin과 /usr/sbin에서 파일이름이 i로 시작하는 것이 몇 개인지 출력하시오.**

grep (regular expression, 정규표현식)

- **grep - 연습문제 답안**

```
joonhee@joonhee-laptop:~$ grep -i '^.j.k' /usr/share/dict/words
Djakarta
Djakarta's
```

```
joonhee@joonhee-laptop:~$ ls /usr/bin | grep gcc
c89-gcc
c99-gcc
```

```
joonhee@joonhee-laptop:~$ ls /usr/bin /usr/sbin | grep '^i' -c
100
```

grep (regular expression, 정규표현식)

- **Metacharacters** (BRE: ^ $. [ ] - * ₩ ERE(-E 옵션필요): ( ) | { } ? +)
  - **Bracket Expressions : [ ]**
    : 지정한 문자 집합 중 하나를 표현
    - 특수문자 효력상실, 예외 두가지

```
✔ ~/os_prac/07 [main|✚1…42]
16:00 $ ls /bin > ls-bin.txt
✔ ~/os_prac/07 [main|✚1…42]
16:00 $ ls /sbin > ls-sbin.txt
```

```
16:00 $ grep -h '[bg]zip' ls*
bzip2
bzip2recover
gzip
```

    - **negation (^, caret)**
      - 괄호 안에서 맨 첫 문자로 오면, 부정형
      - e.g. [^bg]zip : b, g 이외의 다른 문자로 시작하고 zip으로 끝나는 글자 4의 문자열 포함하는 라인

```
16:00 $ grep -h '[^bg]zip' ls*
bunzip2
funzip
gpg-zip
gunzip
mzip
preunzip
prezip
prezip-bin
unzip
unzipsfx
```

      - 첫 문자가 아니면, 일반 문자로 취급

```
✔ ~/os_prac/07 [main|✚1…42]
02:02 $ > b^g
✔ ~/os_prac/07 [main|✚1…42]
02:03 $ ls | grep -E [z^]
b^g
```

grep (regular expression, 정규표현식)

- **Metacharacters** (BRE: ^ $. [ ] - * ₩ ERE(-E 옵션필요): ( ) | { } ? +)
  - **Bracket Expressions : [ ]**
    - **: 지정한 문자 집합 중 하나를 표현**
    - **- 특수문자 효력상실, 예외 두가지**
    - **character range (-, dash)**
      - ✓ **괄호 안에서 문자와 문자사이에 삽입하면 범위를 나타냄**
      - ✓ **e.g. [a-z] : 소문자 하나 *[[:lower:]]*와 동일**

참고: POSIX Character Classes

-, A, Z로 시작하는 라인 없음

```
16:15 $ grep -h '^[-AZ]' ls*
```

```
16:12 $ grep -h '^[BCDAEFGHIZKLMNOPQRSTUVWZYX]' ls*
CartConvert
ConicProj
GeoConvert
```

대문자로 시작하는 라인(복잡)

```
16:12 $ grep -h '^[A-Z]' ls*
CartConvert
ConicProj
GeoConvert
```

대문자로 시작하는 라인(간단)

- ✓ **괄호 안에서 맨 처음 또는 마지막에 오면 일반 문자 취급**

```
16:14 $ grep -h '^[-RM]' ls*
MagneticField
RhumbSolve
ModemManager
```

```
✔ ~/os_prac/07 [main|✚ 1…42]
16:15 $ echo "-joonhee" >> ls-bin.txt
✔ ~/os_prac/07 [main|✚ 1…42]
16:15 $ grep -h '^[-RM]' ls*
MagneticField
RhumbSolve
-joonhee
ModemManager
```

```
02:05 $ grep -h '^[RM-]' ls*
MagneticField
RhumbSolve
-joonhee
ModemManager
```

R, M, -로 시작하는 라인

grep (regular expression, 정규표현식)

- **Metacharacters** (BRE: ^ $. [ ] - * ₩ ERE(-E 옵션필요): ( ) | { } ? +)
  - **Bracket Expressions : [ ]**
    : 지정한 문자 집합 중 하나를 표현
    - 특수문자 효력상실, 예외 두가지
    - **negation (^, caret)**
    - **character range (-, dash)**

간단 연습문제 : 현재 작업 디렉토리의 파일들 중에서 소문자로 시작하지 않는 파일의 목록만을 출력하는 명령어는?

답은 바로 이 밑에 흰색으로 숨겨놓았습니다~

grep (regular expression, 정규표현식)

- **Metacharacters** (BRE: ^ $. [ ] - * ₩ ERE(-E 옵션필요): ( ) | { } ? +)
  - **Alternation (|, bar)**
    : 여러 표현 집합 중에서 하나만 매칭되어도 만족 (or 연산이랑 비슷함)
    e.g. 'AB|CD' AB라는 문자열 또는 CD라는 문자열을 포함하는 라인

```
✔ ~/os_prac/07 [main|✚1…42]
16:36 $ echo "AA" | grep -E 'AA|BB'
AA
✔ ~/os_prac/07 [main|✚1…42]
16:36 $ echo "BB" | grep -E 'AA|BB'
BB
✔ ~/os_prac/07 [main|✚1…42]
16:36 $ echo "CC" | grep -E 'AA|BB'
✘-1 ~/os_prac/07 [main|✚1…42]
```

  - **Combine (( ), parentheses)**
    : 다른 특수기호를 섞어서 사용할 때 씀

```
16:41 $ ls /*bin | grep -E '^(bz|gz|zip)'
```

bz 또는 gz 또는 zip으로 시작하는 라인

```
16:41 $ ls /*bin | grep -E '^bz|gz|zip'
```

bz로 시작하거나 gz 또는 zip을 포함하는 라인

grep (regular expression, 정규표현식)

- **Metacharacters** (BRE: ^ $. [ ] - * ₩ ERE(-E 옵션필요): ( ) | { } ? +)
  - **Quantifiers**
    - **: 원소의 양을 정의**
    - **? : 앞의 원소가 없거나 단 하나 있을 때 매칭**

```
16:41 $ echo "(042) 821-7726" | grep -E '\(?[0-9][0-9][0-9]\)? [0-9]
[0-9][0-9]-[0-9][0-9][0-9][0-9]'
(042) 821-7726
```

```
16:48 $ echo "042 821-7726" | grep -E '\(?[0-9][0-9][0-9]\)? [0-9][0
-9][0-9]-[0-9][0-9][0-9][0-9]'
042 821-7726
```

₩(? 여는 괄호 없거나 있거나
₩)? 닫는 괄호 없거나 있거나
(괄호 특수문자라 backslash)

있을 때, 없을 때 둘 다 매칭!

```
16:50 $ echo "AB2 821-7726" | grep -E '\(?[0-9][0-9][0-9]\)? [0-9][0
-9][0-9]-[0-9][0-9][0-9][0-9]'
```

숫자가 아니라 매칭 실패

grep (regular expression, 정규표현식)

- **Metacharacters** (BRE: ^ $. [ ] - * ₩ ERE(-E 옵션필요): ( ) | { } ? +)
  - **Quantifiers**
    - **: 원소의 양을 정의**
    - *** : 앞의 원소가 임의의 개수가 있을 때 매칭**

```
01:07 $ echo "This works." | grep -E '[[:upper:]][[:upper:][:lower:] ]*\.'
This works.
```

```
01:10 $ echo "OSprac.txt" | grep -E '[[:upper:]][[:upper:][:lower:] ]*\.'
OSprac.txt
```

대문자로 시작하고,
임의 개수만큼 [대소문자, space]가 이어지고,
직후 .(점)으로 이어지는
문자열을 포함하는 라인

```
01:10 $ echo "this does not" | grep -E '[[:upper:]][[:upper:][:lower:] ]*\.'
```

대문자로 시작하지 않아서 실패

grep (regular expression, 정규표현식)

- **Metacharacters** (BRE: ^ $. [ ] - * ₩ ERE(-E 옵션필요): ( ) | { } ? +)
  - **Quantifiers**
    : 원소의 양을 정의
    - + : 앞의 원소가 하나 이상 있을 때 매칭

```
01:11 $ echo "This that" | grep -E '^([[:alpha:]]+ ?)+$'
This that
```

```
01:23 $ echo "a b c" | grep -E '^([[:alpha:]]+ ?)+$'
a b c
```

하나 이상의 알파벳 직후에 space가 있거나 없는,
그런 종류의 문자열이 하나 이상 존재하는데,
그런 종류의 문자열로 시작하고, 끝나는 라인

```
01:28 $ echo "a b 7" | grep -E '^([[:alpha:]]+ ?)+$'
```

마지막이 숫자로 끝나서
(알파벳 또는 space가 아니라서)
실패

grep (regular expression, 정규표현식)

- **Metacharacters** (BRE: ^ $. [ ] - * ₩ ERE(-E 옵션필요): ( ) | { } ? +)
  - **Quantifiers**
    : 원소의 양을 정의
    - {} : 앞의 원소가 지정 개수만큼 있을 때 매칭
      - {n} 정확히 n개
      - {n,m} n개 이상, m개 이하
      - {n,} n개 이상
      - {,m} m개 이하

```
01:41 $ echo "555 123-4567" | grep -E '^\(?[0-9]{3}\)? [0-9]{3}
-[0-9]{4}$'
555 123-4567
```

```
01:41 $ echo "(555) 123-4567" | grep -E '^\(?[0-9]{3}\)? [0-9]{3}
-[0-9]{4}$'
(555) 123-4567
```

시작, 괄호가 있거나, 없거나, 그 사이에 숫자가 3개
한 칸 띄우고, 숫자3개-숫자4개, 끝

```
01:41 $ echo "5555 123-4567" | grep -E '^\(?[0-9]{3}\)? [0-9]{3}
-[0-9]{4}$'
```

숫자 4개로 시작해서 실패

grep (regular expression, 정규표현식)

- **Metacharacters** (BRE: ^ $. [ ] - * ₩ ERE(-E 옵션필요): ( ) | { } ? +)
  - **예제**

```
01:39 $ for i in {1..10}; do echo "(${RANDOM:0:3}) ${RANDOM:0:3}-
${RANDOM:0:4}" >> phonelist.txt; done
```

```
01:43 $ cat phonelist.txt
(156) 257-2060
(179) 877-1120
(303) 298-2851
(255) 706-1558
(120) 207-2634
(266) 191-481
(305) 243-2838
(195) 294-497
(111) 132-1039
(151) 159-2989
```

유사 핸드폰 번호 문자열 생성
(RANDOM? 이해를 돕기 위한 보조 명령어라
자세한 건 저도 잘 몰라요.. 궁금하면 검색 ㄱㄱ)

```
01:43 $ grep -Ev '^\([0-9]{3}\) [0-9]{3}-[0-9]{4}$' phonelist.txt
(266) 191-481
(195) 294-497
```

(???) ???-???? 또는 ??? ???-???? 패턴에
불일치(-v)하는 라인 출력
(뒷 번호가 네 자리가 아닌 세 자리)

동시성 문제 해결

1. atomicity
2. ordering
3. deadlock

Concurrency

- **pthread.h : thread 관련 라이브러리**
  - **컴파일시 -pthread 옵션 필수** (gcc test.c -o test **-pthread)**
  - **선언 헤더 #include <pthread.h>**

함수명	설명
int pthread_create (pthread_t *restrict *thread*, const pthread_attr_t *restrict *attr*, void *(**start_routine*)(void *), void *restrict *arg*);	- 새로운 쓰레드를 생성 - 스케쥴러에 의해 실행 쓰레드 결정 pthread_t: thread 식별자 restrict: 해당 메모리 유일접근가능 포인터 쓰레드 저장할 포인터 / 쓰레드 설정 포인터 / 쓰레드 실행루틴 / 실행루틴 인자 성공시 0, 실패시 에러넘버 리턴(join 동일)
int pthread_join(pthread_t *thread*, void **retval*)	- 쓰레드 종료 대기 - 이미 종료된 경우 즉시 리턴 종료 대기 대상 쓰레드 / 실행루틴 종료 상태 저장 포인터
pthread_t pthread_self(void)	- 실행 쓰레드 id 리턴

Concurrency

■ pthread.h : thread 관련 라이브러리

함수/변수명	설명
int pthread_mutex_lock(pthread_mutex_t *mutex);	- lock 획득 - 누군가 lock을 갖고 있으면, 그 친구가 unlock해서 lock을 획득할 때까지 block 성공시 0, 실패시 에러 넘버 리턴 (이하 동일)
int pthread_mutex_unlock(pthread_mutex_t *mutex);	- lock 해제
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;	- 초기화 상수 - pthread_mutex_init(&mutex, NULL)과 동일

Concurrency

■ pthread.h : thread 관련 라이브러리

함수/변수명	설명
int pthread_cond_signal(pthread_cond_t *cond);	- 인자의 조건변수에 대해 (block된 쓰레드가 있다면) block된 쓰레드 중 최소 하나를 unblock 성공시 0, 실패시 에러 넘버 리턴 (이하 동일)
int pthread_cond_wait(pthread_cond_t restrict*cond, pthread_mutex_t restrict*mutex);	- 인자의 조건변수에 대해 호출한 쓰레드를 block - 호출시 lock을 갖고 있을 것을 전제 - atomically, lock을 release하고, sleep
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;	- 초기화 상수

Concurrency

- **Semaphore**
  - **#include <semaphore.h>**
  - **컴파일시 -phread 옵션 필수**

Semaphore 관련 함수	설명
sem_t *sem_open(const char* name, int oflag, mode_t mode, int value)	Named 세마포어 생성
sem_unlink(const char* name)	Named 세마포어 제거
sem_close(sem_t* sem)	Named 세마포어 사용 종료
int sem_init(sem_t* sem, int pshared, unsigned value)	세마포어 할당
sem_destroy(sem_t* sem)	세마포어 반환
int sem_wait(sem_t* sem)	세마포어 lock
int sem_post(sem_t* sem)	세마포어 unlock

동시성 문제

- **atomicity**

```
37 void *thread1(void *arg) {
38     printf("t1: before check\n");
39     if (thd->proc_info) {
40         printf("t1: after check\n");
41         sleep(2);
42         printf("t1: use!\n");
43         printf("%d\n", thd->proc_info->pid);
44     }
45     return NULL;
46 }
47
48 void *thread2(void *arg) {
49     printf("                    t2: begin\n");
50     sleep(1); // change to 5 to make the code "work"...
51     printf("                    t2: set to NULL\n");
52     thd->proc_info = NULL;
53     return NULL;
54 }
55
```

결과는 다음 슬라이드..

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <pthread.h>
 4 #include <unistd.h>
 5
 6 typedef struct {
 7     int pid;
 8 } proc_t;
 9
10 typedef struct {
11     proc_t *proc_info;
12 } thread_info_t;
13
14 proc_t p;
15 thread_info_t *thd;
16
17 void *thread1(void *arg);
18 void *thread2(void *arg);
19
20 int main(int argc, char *argv[]) {
21     thread_info_t t;
22     p.pid = 100;
23     t.proc_info = &p;
24     thd = &t;
25
26     pthread_t p1, p2;
27     printf("main: begin\n");
28     pthread_create(&p1, NULL, thread1, NULL);
29     pthread_create(&p2, NULL, thread2, NULL);
30     // join waits for the threads to finish
31     pthread_join(p1, NULL);
32     pthread_join(p2, NULL);
33     printf("main: end\n");
34     return 0;
35 }
```

동시성 문제

- **atomicity**

```
04:48 $ ./atomicity
main: begin
t1: before check
t1: after check
                t2: begin
                t2: set to NULL
t1: use!
Segmentation fault (core dumped)
```

오답
(주어진 코드)

```
05:16 $ ./atomicity_answer
main: begin
                t2: begin
t1: before check
t1: after check
t1: use!
100
                t2: set to NULL
main: end
```

정답
(목적 코드)

1. 문제 원인 분석

2. 지금까지 배운 도구를 자유롭게 활용하여 문제 해결 (코드는 추가만 하고 제거는 하지 말 것)

동시성 문제

- **ordering**

```
05:06 $ ./ordering
ordering: begin
routine: begin
Segmentation fault (core dumped)
```

오답
(주어진 코드)

```
05:10 $ ./ordering_answer
ordering: begin
routine: begin
routine: state is 0
ordering: end
```

정답
(목적 코드)

1. 문제 원인 분석

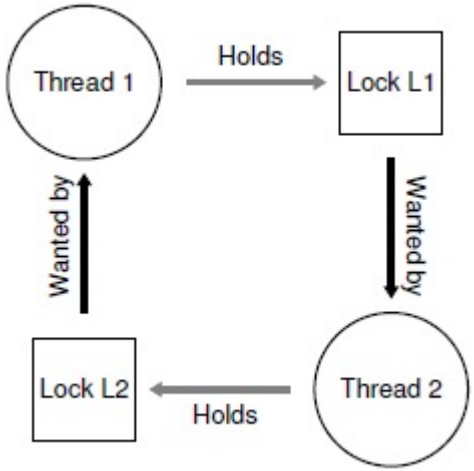
2. 지금까지 배운 도구를 자유롭게 활용하여 문제 해결 (코드는 추가만 하고 제거는 하지 말 것)

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

#define STATE_INIT (0)

typedef struct {
    pthread_t th;
    int state;
} my_thread_t;

my_thread_t *thd;

void *routine(void *arg) {
    printf("routine: begin\n");
    printf("routine: state is %d\n", thd->state);
    return NULL;
}

void myWaitThread(my_thread_t *p) {
    pthread_join(p->th, NULL);
}

my_thread_t *myCreateThread(void *(*start_routine)(void *)) {
    my_thread_t *p = malloc(sizeof(my_thread_t));
    if (p == NULL)
        return NULL;
    p->state = STATE_INIT;
    pthread_create(&p->th, NULL, start_routine, NULL);
    // turn the sleep off to avoid the fault, sometimes...
    sleep(1);
    return p;
}

int main(int argc, char *argv[]) {
    printf("ordering: begin\n");
    thd = myCreateThread(routine);
    myWaitThread(thd);
    printf("ordering: end\n");
    return 0;
}
```

동시성 문제

deadlock

```
Thread 1:                    Thread 2:
pthread_mutex_lock(L1);      pthread_mutex_lock(L2);
pthread_mutex_lock(L2);      pthread_mutex_lock(L1);
```




```
05:27 $ for i in {1..10000}; do ./deadlock; done
```

for i in {1..10000}; do ./deadlock; done

실행 해보면 데드락 발생할 것임.
결과는 다음 슬라이드..

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

pthread_mutex_t L1 = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t L2 = PTHREAD_MUTEX_INITIALIZER;

void *thread1(void *arg) {
    printf("t1: begin\n");
    printf("t1: try to acquire L1...\n");
    pthread_mutex_lock(&L1);
    printf("t1: L1 acquired\n");
    printf("t1: try to acquire L2...\n");
    pthread_mutex_lock(&L2);
    printf("t1: L2 acquired\n");
    pthread_mutex_unlock(&L1);
    pthread_mutex_unlock(&L2);
    return NULL;
}

void *thread2(void *arg) {
    printf("                              t2: begin\n");
    printf("                              t2: try to acquire L2...\n");
    pthread_mutex_lock(&L2);
    printf("                              t2: L2 acquired\n");
    printf("                              t2: try to acquire L1...\n");
    pthread_mutex_lock(&L1);
    printf("                              t2: L1 acquired\n");
    pthread_mutex_unlock(&L1);
    pthread_mutex_unlock(&L2);
    return NULL;
}

int main(int argc, char *argv[]) {
    pthread_t p1, p2;
    printf("main: begin\n");
    pthread_create(&p1, NULL, thread1, NULL);
    pthread_create(&p2, NULL, thread2, NULL);
    // join waits for the threads to finish
    pthread_join(p1, NULL);
    pthread_join(p2, NULL);
    printf("main: end\n");
    return 0;
}
```

동시성 문제

- **deadlock**

```
main: begin
t1: begin
t1: try to acquire L1...
t1: L1 acquired
                          t2: begin
                          t2: try to acquire L2...
                          t2: L2 acquired
                          t2: try to acquire L1...
t1: try to acquire L2...
^C
```

데드락 발생 Case
(Ctrl+C Interrupt 발생시켜서 종료)

```
main: begin
                          t2: begin
                          t2: try to acquire L2...
                          t2: L2 acquired
                          t2: try to acquire L1...
                          t2: L1 acquired
t1: begin
t1: try to acquire L1...
t1: L1 acquired
t1: try to acquire L2...
t1: L2 acquired
main: end
```

데드락 미발생 Case

Q. 데드락이 발생하지 않도록 어떻게 방지할까?

데드락 발생 조건 4가지 -> *네 가지 중 한 가지만이라도 깨버리자!*

Mutual Exclusion (상호 배제)
- 자원을 공유 가능하게 한다. ➔ 원천적으로 불가능

Hold and wait (점유 후 대기)
- 모든 자원을 가지든, 아예 가지지 않도록 만든다. ➔ 가능하지만, 자원 활용률이 안 좋음

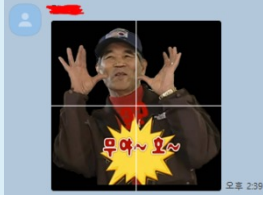
No Preemption(비 선점)
- 점유한 자원을 내놓게 한다. ➔ 가능하지만, 자원 활용률 안 좋음

Circular wait(환형 대기)
- 자원 할당 시 순서를 부여하여 원형의 모양을 깬다. ➔ 가능하지만 불편 (리눅스 커널 구현에서 이 방법을 씀)

과제

1. atomicity / ordering problem
2. deadlock prevention

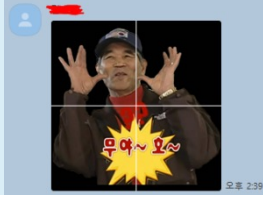
과제



https://github.com/JoonheeJeong/cnu_os_prac/tree/main/07

git fetch-full해서 쓰시거나 그냥 복붙하셔요~

과제

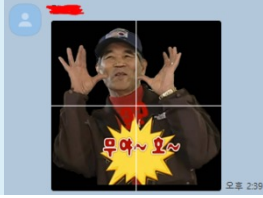


- atomicity / ordering problem
  - 뮤텍스, 조건변수, 세마포어 등 지금까지 배운 동시성 제어 라이브러리를 활용하여 목적 결과를 도출하는 코드 작성
  - 코드는 주어진 코드에서 추가만 하면 되고, 따로 제거할 필요 없음 (제거 하지 말 것)
  - 소스 코드, 실행 결과, 코드 설명 및 결과 분석

빨간색 필수
초록색 가산점

- deadlock prevention
  - Circular wait, Hold-and-wait, No preemption 세 가지 속성에 대하여 각 속성을 깨뜨리는(break) 방법을 검색해보고 보고서 작성
  - 구체적으로 코드 관점에서 어떤 식으로 바뀌어야 하는지 찾아볼 것
  - 찾아 본 내용을 바탕으로 각각의 속성을 깨뜨리는 버전의 코드를 직접 작성하여 코드와 실행결과를 보여주면 매우 훌륭함 (가산점)
  - 방법 (왜 작성한 방법이 유효한지 구체적인 설명), 소스 코드, 실행 결과, 코드 설명 및 결과 분석

과제



포함 내용

- 소스코드 (src 폴더에 담아주시면 감사합니다.)
  - atomicity_학번.c
  - ordering_학번.c
  - circular_wait_학번.c / hold_and_wait_학번.c / no_preemption_학번.c
- 보고서
  - 코드 이미지 및 결과 이미지
  - 코드와 결과 부분부분 대응하여 설명
  - deadlock 방법 / 소스코드를 작성했다면 바로 위 두 줄처럼 작성
  - 느낀 점 (쪽지시험, 수업, 과제 재미, 난이도 기타 등등)
  - OS01_07_학번_이름.pdf
    - ✓ 맥북 유저분들, pdf인지 확인하세요!

빨간색 필수
초록색 가산점

제출기한 및 양식

- ~5월 18일(화) 23:59 (2주) (다음 주에 과제 안 낼게요. 하하하~)
- OS01_07_학번_이름.zip
  - 소스코드
  - 보고서

Q&A

Thank you!!!