운영체제및실습 01분반
실습 5회차

조건변수

2021.04.14(수)

Contents

리눅스 명령어(4)

1. history
2. bash Expansion/Quoting 복습
3. grep (regular expression) (1)

리눅스 명령어(4)

history

: 명령어 입력 기록

*기본
위 화살표 - 역순 이동
아래 화살표 - 순 이동

*유의 사항
- history list의 현재 item이 계속 이동됨
(마우스 커서 마냥 포인터가 이동한다고 생각하세요!

- 기본적으로 맨 마지막에 포인터가 있음
- enter를 눌러서 명령어를 입력하면 포인터가 다시 맨 마지막으로 이동

~/.bash_history 파일에 명령어 입력 기록 저장

리눅스 명령어(4)

history

history : 화면에 명령어 입력 기록 출력
!number : 명령어 수행

ctrl-r : 역순 검색

```
joonhee@joonhee-VM:~$
(reverse-i-search)`':
```

계속 ctrl-r 반복하면 거슬러 올라감

```
(reverse-i-search)`ls -l': ls -l /usr/bin > ls-output.txt
```

ctrl-j : 명령어 복사 | **enter** : 명령어 수행

```
joonhee@joonhee-VM:~$ ls -l /usr/bin > ls-output.txt
```

(**ctrl-r 빠져나오고 싶으면**) ctrl-c or ctrl-g

```
joonhee@joonhee-VM:~$ history | less
joonhee@joonhee-VM:~$ history | grep /usr/bin
   45  sudo cp test /usr/bin/정준희
   63  sudo cp test /usr/bin/정준희
   64  sudo cp test /usr/bin/정준희 | echo
   65  echo "sudo cp test /usr/bin/정준희"
   76  rm /usr/bin/정준희
   77  sudo rm /usr/bin/정준희
   85  ls /usr/bin/정준희
   86  sudo ls /usr/bin/정준희
   87  sudo ls /usr/bin
   95  ls -l /usr/bin > ls-output.txt
   97  history | grep /usr/bin
joonhee@joonhee-VM:~$ rm ls-output.txt
joonhee@joonhee-VM:~$ !95
ls -l /usr/bin > ls-output.txt
joonhee@joonhee-VM:~$ less ls-output.txt
```

리눅스 명령어(4)

■ history (참고)

alt-p : 역순 검색 (반복 없음, 포인터 유지)
alt-n : 순 검색 (반복 없음, 포인터 유지)

- enter 누르면 해당 명령어 복사
- 동일한 입력, 동일한 결과

alt-< : 포인터 시작점으로 이동 (alt+shift+,)
alt-> : 포인터 끝점으로 이동 (default 위치)

- 이후에 위/아래 화살표 움직여보면 차이를 알 수 있음

ctrl-o : 현재 명령줄 내용 입력 후 포인터 다음으로 이동 후 명령줄로 fetch

◆ history expansion

history 명령어로 번호 확인 후 수행하기
!자칫 의도하지 않은 수행 발생!

!! : 마지막 실행 명령어 수행
!number : 번호 해당 명령어 수행
!string : 입력 문자열로 시작하는 마지막 명령어 수행
!?string : 입력 문자열을 포함하는 마지막 명령어 수행

```
joonhee@joonhee-vm:~$ ls -l /usr/bin | grep zip
joonhee@joonhee-vm:~$ !!
ls -l /usr/bin | grep zip
-rwxr-xr-x 1 root root       39144  9월  6  2019 bunzip2
joonhee@joonhee-vm:~$ !ls
ls -l /usr/bin | grep zip
-rwxr-xr-x 1 root root       39144  9월  6  2019 bunzip2
joonhee@joonhee-vm:~$ !?bin
ls -l /usr/bin | grep zip
-rwxr-xr-x 1 root root       39144  9월  6  2019 bunzip2
```

```
1163  ls -l /usr/bin | grep zip
1164  history | less
(END)
joonhee@joonhee-vm:~$ !1163
ls -l /usr/bin | grep zip
```

리눅스 명령어(4)

bash Expansion/Quoting 복습 (연습문제)

1. arithmetic expansion을 활용하여 echo로 3*(5+7)의 결과를 출력하는 명령줄은?
2. ls 명령어의 파일 타입을 확인하기 위한 한 줄 명령어는? (hint: file, which)
3. echo {b..d}_{d..b}의 결과는?
4. echo {b..d} {d..b}의 결과는? (위 문제와 다릅니다.)
5. 현재 작업 디렉토리와 그 부모 디렉토리를 제외하고, 현재 작업 디렉토리 내 숨김 파일을 포함한 모든 파일을 표시하는 한 줄 명령어는? (echo로 시작하세요)
6. 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(4)

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

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

```
09:30 $ echo {b..d} {d..b}
b c d d c b
```

```
09:29 $ echo {b..d}_{d..b}
b_d b_c b_b c_d c_c c_b d_d d_c d_b
```

```
09:30 $ file $(which ls)
/usr/bin/ls: ELF 64-bit LSB shared object, x
```

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

```
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
```

리눅스 명령어(4)

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
```

리눅스 명령어(4)

■ 정규표현식 (regular expression)

: 문자열을 나타내는 패턴

- 대표적으로 두 종류의 정규표현식 존재

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

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

shell에 인자로 넘길 때 expansion 고려해서 적절히 quoting해야 함

일반적으로 싱글 쿼팅이 expansion을 완전 차단하므로 더 추천!

- . : 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
gzip
```

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

```
07:42 $ ls -l /usr/bin | grep "zip$"
-rwxr-xr-x 1 root root       26776  8월 16  2019 funzip
-rwxr-xr-x 1 root root        3516  1월  7 03:10 gpg-zip
```

```
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
```

리눅스 명령어(4)

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

리눅스 명령어(4)

- **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
```

CONDITION VARIABLES

1. create, join, self
2. lock/unlock, init/destory

Condition Variables

- 조건 변수 (Condition Variables)
  - 쓰레드간 상태의 변화(조건의 만족 여부)를 알리는 방법이 없을까?
    - 작업의 선수조건 (1. 조건 만족시 수행, 2. 반드시 수행)

```
 4 volatile int counter = 0;
 5 const int limit  = 1000000000; // 10억
 6 const int signal = 100000000;  // 1억
 7
 8 void *child(void *arg)
 9 {
10     printf("child: begin\n");
11     for (int i = 0; i < limit; i++)
12         counter++;
13     printf("child: done\n");
14     return NULL;
15 }
16
17 int main()
18 {
19     printf("parent: begin\n");
20     pthread_t c;
21     pthread_create(&c, NULL, child, NULL);
22  이후 parent의 수행에서,
23  counter가 반드시 1억(signal) 이상이어야 한다면?
24  어떤 코드를 넣어야 할까?
25     printf("parent: done\n");
26     return 0;
27 }
approach.c" [Modified] 27 lines --11%--
```

Condition Variables

■ 조건 변수 (Condition Variables)

- ◆ 쓰레드간 상태의 변화(조건의 만족 여부)를 알리는 방법이 없을까?

- ◆ 간단한 해결책

: busy waiting(spin)

- ◆ 해당 방법의 단점

-> CPU 낭비

```
 4 volatile int counter = 0;
 5 const int limit  = 1000000000; // 10억
 6 const int signal = 100000000;  // 1억
 7
 8 void *child(void *arg)
 9 {
10     printf("child: begin\n");
11     for (int i = 0; i < limit; i++)
12         counter++;
13     printf("child: done\n");
14     return NULL;
15 }
16
17 int main()
18 {
19     printf("parent: begin\n");
20     pthread_t c;
21     pthread_create(&c, NULL, child, NULL);
22     while (counter < signal) {
23         ; // spin
24     }
25     printf("parent: done\n");
26     return 0;
27 }
approach.c" [Modified] 27 lines --11%--
```

Condition Variables

■ 조건 변수 (Condition Variables)

- spin의 해결책 : 조건 변수 활용!

```
 7 pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;
 8 pthread_cond_t c = PTHREAD_COND_INITIALIZER;
 9
10 void *child(void *arg)
11 {
12     printf("child: begin\n");
13     for (int i = 0; i < limit; i++) {
14         pthread_mutex_lock(&m);
15         counter++;
16         if (counter >= signal)
17             pthread_cond_signal(&c);
18         pthread_mutex_unlock(&m);
19     }
20     printf("child: done\n");
21     return NULL;
22 }
23
24 void my_wait()
25 {
26     pthread_mutex_lock(&m);
27     while (counter < signal)
28         pthread_cond_wait(&c, &m);
29     pthread_mutex_unlock(&m);
30 }
31
32 int main()
33 {
34     printf("parent: begin\n");
35     pthread_t c;
36     pthread_create(&c, NULL, child, NULL);
37     my_wait();
38     printf("parent: done\n");
39     return 0;
40 }
approach_v2.c" 40 lines --77%--
```

- lock이 없으면?
  - 조건 검사 후, wait 호출 직전에 interrupt 되어서 child가 종료되면, 깨워줄 다른 쓰레드가 없다ㅜㅜ!
- 조건 검사가 없으면?
  - child가 먼저 수행된다면, wait후에 signal로 깨워줄 다른 쓰레드가 없다ㅜㅜ!

Condition Variables

- **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

Condition Variables

■ 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

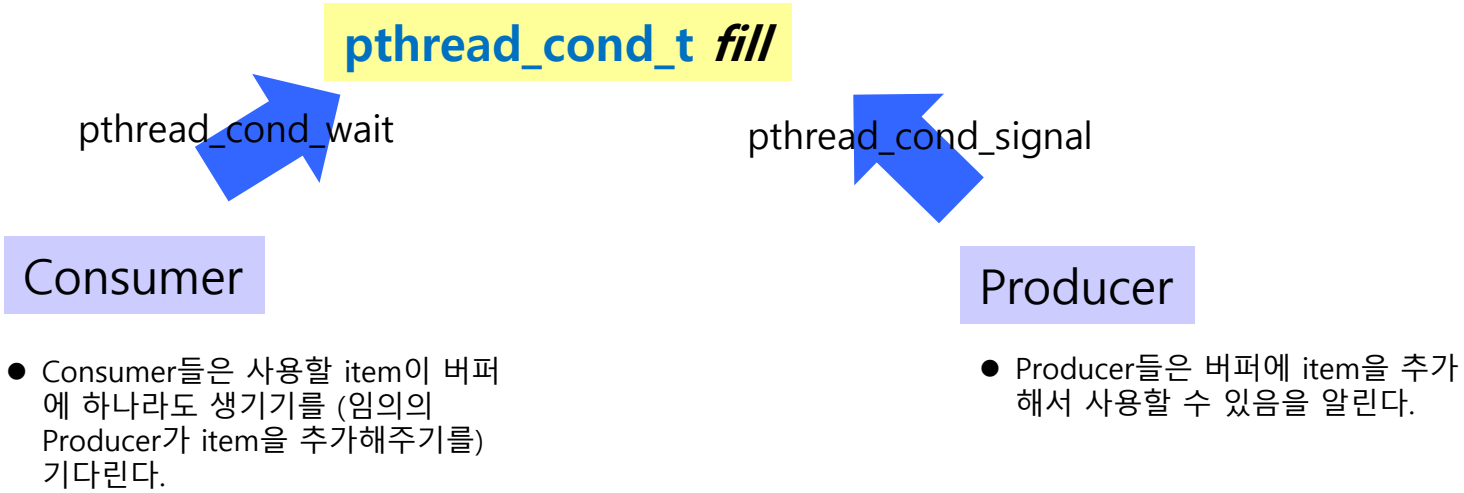
Condition Variables

■ 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

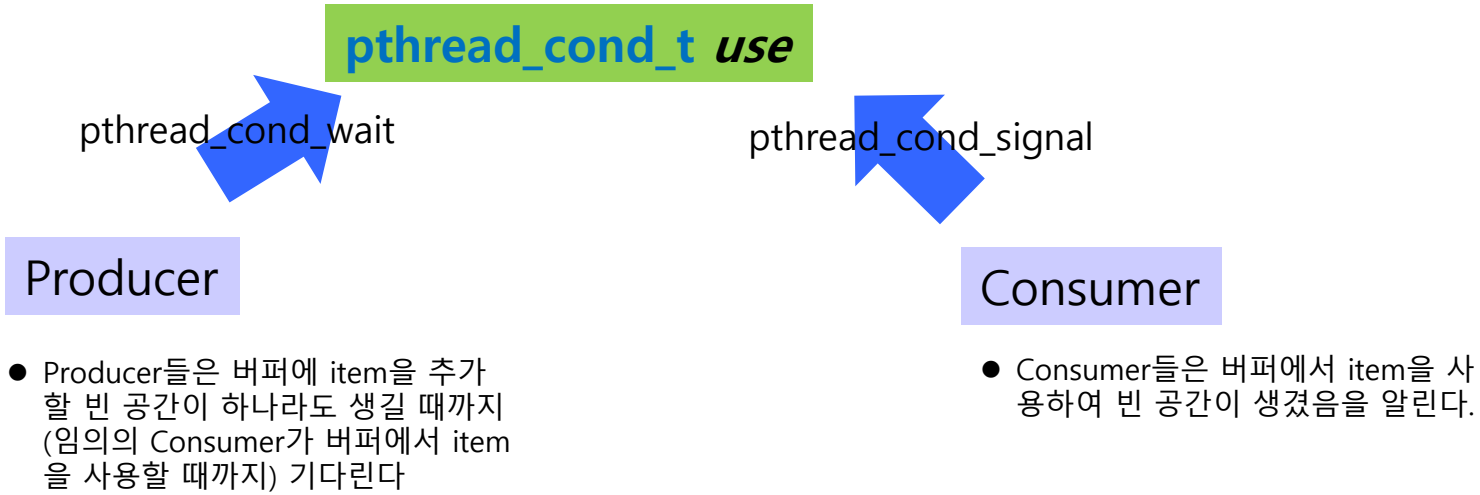
Condition Variables

- **Producer-Consumer Problem**
  - **mutex를 사용하여 공유 변수 접근 제어**
  - **signal을 사용하여 buffer 변화 감지**
    - **buffer에 item을 추가했다는 것을 알리는 조건변수**



Condition Variables

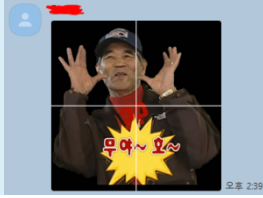
- **Producer-Consumer Problem**
  - **mutex를 사용하여 공유 변수 접근 제어**
  - **signal을 사용하여 buffer 변화 감지**
    - **buffer에서 item을 사용했다는 것을 알리는 조건변수**



과제

1. producer-consumer problem with condition variables

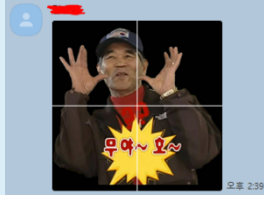
과제



https://github.com/JoonheeJeong/cnu_os_prac/blob/main/05/pc_cv_homework.c

git fetch-full해서 쓰시거나 그냥 복붙하세요~

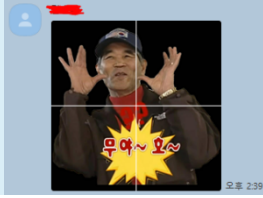
과제



```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>     // usleep (micro sleep)

#define MAX 10
#define PROD_SIZE 3
#define CONS_SIZE 7
#define PROD_ITEM 5
#define CONS_ITEM 2

/* homework */
/*----------*/
/* You need to make some variables (mutex, condition variables). */
/* Reference below variables, functions. */
/*----------*/
/* homework */

pthread_mutex_t m_id = PTHREAD_MUTEX_INITIALIZER;
int buffer[MAX]; // ! buffer is circular queue !
int count = 0;
int get_ptr = 0;
int put_ptr = 0;
int prod_id = 1;
int cons_id = 1;
```

```c
int main()
{
    pthread_t prod[PROD_SIZE];
    pthread_t cons[CONS_SIZE];

    for (int i = 0; i < PROD_SIZE; ++i)
        pthread_create(&prod[i], NULL, producer, NULL);
    for (int i = 0; i < CONS_SIZE; ++i)
        pthread_create(&cons[i], NULL, consumer, NULL);

    for (int i = 0; i < PROD_SIZE; ++i)
        pthread_join(prod[i], NULL);
    for (int i = 0; i < CONS_SIZE; ++i)
        pthread_join(cons[i], NULL);

    return 0;
}
```

과제

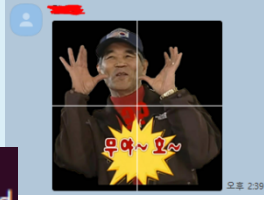


```c
void *producer(void *arg)
{
    pthread_mutex_lock(&m_id);
    int id = prod_id++;
    pthread_mutex_unlock(&m_id);
    for (int i = 0; i < PROD_ITEM; ++i) {
        usleep(10);

        /* homework */
        /*----------*/
        int ret = put(i);
        /*----------*/
        /* homework */

        if (ret == -1) {
            printf("can't put, becuase buffer is full.\n");
        } else {
            printf("producer %d PUT %d\n", id, ret);
        }
    }
}

```

```c
void *consumer(void *arg)
{
    pthread_mutex_lock(&m_id);
    int id = cons_id++;
    pthread_mutex_unlock(&m_id);
    for (int i = 0; i < CONS_ITEM; ++i) {
        usleep(10);

        /* homework */
        /*----------*/
        int ret = get();
        /*----------*/
        /* homework */

        if (ret == -1) {
            printf("can't get, becuase buffer is empty.\n");
        } else {
            printf("consumer %d GET %d\n", id, ret);
        }
    }
}

```

```c
/* homework */
// return buffer's value using get_ptr if successful,
// otherwise, -1
int get()
{
    return -1;
}

/* homework */
// return buffer's value using put_ptr if successful,
// otherwise, -1
int put(int val)
{
    return -1;
}

```

과제



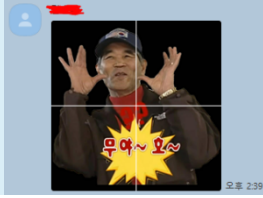
```
✔ ~/os_prac/05 [main|✚ 1…12]
08:48 $ gcc pc_homework.c -o pc_homework -pthread
✔ ~/os_prac/05 [main|✚ 1…12]
08:48 $ ./pc_homework
producer 1 PUT 0
producer 2 PUT 0
consumer 1 GET 0
consumer 2 GET 0
can't get, becuase buffer is empty.
can't get, becuase buffer is empty.
can't get, becuase buffer is empty.
can't get, becuase buffer is empty.
can't get, becuase buffer is empty.
can't get, becuase buffer is empty.
can't get, becuase buffer is empty.
producer 2 PUT 1
producer 1 PUT 1
consumer 3 GET 1
producer 3 PUT 0
producer 2 PUT 2
producer 1 PUT 2
producer 3 PUT 1
producer 2 PUT 3
producer 1 PUT 3
producer 3 PUT 2
consumer 4 GET 1
producer 2 PUT 4
producer 1 PUT 4
producer 3 PUT 3
consumer 5 GET 0
consumer 6 GET 2
consumer 7 GET 2
producer 3 PUT 4
✔ ~/os_prac/05 [main|✚ 1…12]
08:48 $
```

오답 예시

```
✔ ~/os_prac/05 [main|✚ 1…12]
08:34 $ gcc pc.c -o pc -pthread
✔ ~/os_prac/05 [main|✚ 1…12]
08:34 $ ./pc
producer 1 PUT 0
producer 2 PUT 0
consumer 1 GET 0
consumer 2 GET 0
producer 3 PUT 0
consumer 3 GET 0
producer 3 PUT 1
consumer 4 GET 1
producer 3 PUT 2
consumer 6 GET 2
producer 3 PUT 3
consumer 2 GET 3
producer 1 PUT 1
producer 2 PUT 1
consumer 1 GET 1
consumer 3 GET 1
producer 3 PUT 4
consumer 7 GET 4
producer 1 PUT 2
producer 2 PUT 2
producer 1 PUT 3
consumer 7 GET 2
consumer 4 GET 2
consumer 5 GET 3
producer 2 PUT 3
consumer 6 GET 3
producer 1 PUT 4
producer 2 PUT 4
consumer 5 GET 4
✔ ~/os_prac/05 [main|✚ 1…12]
08:34 $
```

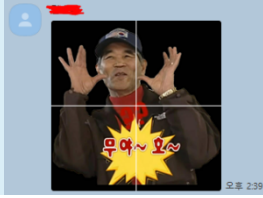
답안 예시

과제



- pthread 변수, get/put, consumer/producer 함수 작성
- 정답과 출력의 모든 순서가 똑같아야 하는 것 아님 (당연히 다를 가능성이 99.9~%)
- 다만 for문의 출력이 정해져서 출력 순서만 다를 뿐 값은 같으므로 확인할 것
  - put = 3 * 5 = 15회 / get = 7 * 2 = 14회 / 총 29회
  - 즉 ./pc | wc -l 의 결과가 29
- pthread 함수(뮤텍스, 조건변수) 외 하드코딩으로 유사 답안 만들면 오답처리
- pthread 함수를 올바르게 사용하지 않았는데 우연히 나온 좋은 결과로 보고서를 쓸 경우 오답처리 (100번 시도하면 100번 성공해야 함)

과제



- 포함 내용
  - 소스코드
    - pc_cv_학번.c
  - 보고서
    - 코드 이미지 및 설명 (그렇게 작성한 이유), 결과 화면
    - 느낀 점 (쪽지시험, 수업, 과제 재미, 난이도 기타 등등)
    - OS01_05_학번_이름.pdf
      - ✓ 맥북 유저분들, pdf인지 확인하세요!

- 제출기한 및 양식
  - ~4월 27일(화) 23:59 (2주, 중간고사 끝나고)
  - OS01_05_학번_이름.zip
    - 소스코드 (src 폴더 필요 없음. 있어도 상관 없음.)
    - 보고서

Q&A

Thank you!!!