운영체제및실습 01분반
실습 4회차

쓰레드 및 뮤텍스

2021.04.07(수)

Contents

리눅스 명령어(3)

1. Command Line Editing
2. Expansion
3. Quoting

리눅스 명령어(3)

(주의!) 셸 구현체마다 결과의 차이가 있을 수 있음 (본 실습은 bash로 진행)

Command Line Editing

ctrl a : 커서 맨 처음으로
ctrl e : 커서 맨 끝으로

alt f : 커서 한 단어 앞으로(오른쪽으로)
alt b : 커서 한 단어 뒤로(왼쪽으로)

alt d : 앞으로 한 단어 잘라내기
ctrl w(alt backspace) : 뒤로 한 단어 잘라내기

ctrl k : 커서부터 맨 끝까지 잘라내기
ctrl u : 커서부터 맨 처음까지 잘라내기

ctrl y : 버퍼 붙여넣기 (앞으로/뒤로 잘라내기 섞으면 버퍼 초기화됨)
alt t : 단어 순서 바꿔치기 (두 단어 사이에서 수행)

alt u : 커서부터 맨 끝까지 대문자로
alt l : 커서부터 맨 끝까지 소문자로

ctrl l : 커서 위치 터미널 맨 위로 (clear는 내용 삭제까지)

리눅스 명령어(3)

- **Expansion**
  : 셸이 사용자의 입력 커맨드를 수행하기 전에 확장 해석하는 방식

  - **Pathname Expansion (<==Wildcards)**

```
joonhee@joonhee-laptop:~$ echo this is a test
this is a test
joonhee@joonhee-laptop:~$ echo *
argos3 argos3-examples catkin_ws Desktop Documents Downloads mavlink Music os_pr
ac ostep Pictures Public snap Templates Videos
joonhee@joonhee-laptop:~$ ls
argos3           Desktop    mavlink  ostep      snap
argos3-examples  Documents  Music    Pictures   Templates
catkin_ws        Downloads  os_prac  Public     Videos
joonhee@joonhee-laptop:~$ echo D*
Desktop Documents Downloads
joonhee@joonhee-laptop:~$ echo *s
argos3-examples catkin_ws Documents Downloads Pictures Templates Videos
joonhee@joonhee-laptop:~$ echo [[:upper:]]*
Desktop Documents Downloads Music Pictures Public Templates Videos
joonhee@joonhee-laptop:~$ echo D*l*
Downloads
```

echo .[!.]* ls -A (almost all)

```
joonhee@joonhee-laptop:~$ echo .*
. .. .bash-git-prompt .bash_history .bash_logout .bashrc .cache .config .gitconf
ig .gnupg .hello_world.c.swp .lesshst .local .profile .python_history .ros .sudo
_as_admin_successful .vboxclient-clipboard.pid .vboxclient-display-svga-x11.pid
.vboxclient-draganddrop.pid .vboxclient-seamless.pid .viminfo .vimrc
```

리눅스 명령어(3)

- **Expansion**
  - **Tilde Expansion (~ 기호, 틸드라고 발음)**

```
joonhee@joonhee-laptop:~$ echo ~
/home/joonhee
```

```
joonhee@joonhee-laptop:~$ echo ~joonhee
/home/joonhee
```

~user

  - **Arithmetic Expansion**

$((*expression*))

숫자 : number 0number(8진수) 0xnumber(16진수) base#number

연산자 : + - * / %(modulo) **(exponent)

```
joonhee@joonhee-laptop:~$ echo $((0xff))
255
joonhee@joonhee-laptop:~$ echo $((2#11111111))
255
```

```
joonhee@joonhee-laptop:~$ echo $(($((5**2)) * 3))
75
joonhee@joonhee-laptop:~$ echo $(((5**2) * 3))
75
```

비트연산자 : ~ << >> & | ^

```
joonhee@joonhee-laptop:~$ echo $(((1<<31)-1)) is int_32 maximum value
2147483647 is int_32 maximum value
```

```
joonhee@joonhee-laptop:~$ echo $((2#1010 ^ 10))
0
```

리눅스 명령어(3)

- **Expansion**
  - **Brace Expansion**

: 다중 결과 출력에 유용

```
joonhee@joonhee-laptop:~$ echo Front-{A,B,C}-Back
Front-A-Back Front-B-Back Front-C-Back
```

```
joonhee@joonhee-laptop:~$ echo Number_{1..5}
Number_1 Number_2 Number_3 Number_4 Number_5
```

zero padding

```
joonhee@joonhee-laptop:~$ echo {001..15}
001 002 003 004 005 006 007 008 009 010 011 012 013 014 015
```

```
joonhee@joonhee-laptop:~$ echo {01..15}
01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
```

```
joonhee@joonhee-laptop:~$ echo {09..1}
09 08 07 06 05 04 03 02 01
```

```
joonhee@joonhee-laptop:~$ echo {a..f}
a b c d e f
```

```
joonhee@joonhee-laptop:~$ echo {z..k}
z y x w v u t s r q p o n m l k
```

역순 가능

nested braeces

```
joonhee@joonhee-laptop:~$ echo a{A{1,2},B{3,4}}b
aA1b aA2b aB3b aB4b
```

```
joonhee@joonhee-laptop:~$ mkdir photos
joonhee@joonhee-laptop:~$ cd photos/
joonhee@joonhee-laptop:~/photos$ mkdir {2019..2020}-{01..12}
joonhee@joonhee-laptop:~/photos$ ls
2019-01  2019-04  2019-07  2019-10  2020-01  2020-04  2020-07  2020-10
2019-02  2019-05  2019-08  2019-11  2020-02  2020-05  2020-08  2020-11
2019-03  2019-06  2019-09  2019-12  2020-03  2020-06  2020-09  2020-12
```

리눅스 명령어(3)

Expansion

Parameter Expansion

```
joonhee@joonhee-laptop:~$ echo $USER
joonhee
```

```
joonhee@joonhee-laptop:~$ echo $greongeo_upda
```

```
joonhee@joonhee-laptop:~$ greongeo_upda=$((0xff))
joonhee@joonhee-laptop:~$ echo $greongeo_upda
255
joonhee@joonhee-laptop:~$ echo $((greongeo_upda++))
255
joonhee@joonhee-laptop:~$ echo $greongeo_upda
256
```

셸 변수 할당
auto completion 활용

- **printenv | less**
  - ✓ 모든 환경변수 확인

Command Substitution

: 커맨드의 출력 활용

```
joonhee@joonhee-laptop:~$ ls -l $(which cp)
-rwxr-xr-x 1 root root 153976  9월  5  2019 /usr/bin/cp
```

```
joonhee@joonhee-laptop:~$ echo $(ls)
argos3 argos3-examples catkin_ws Desktop Documents Downloads mavlink Music os_prac
 ostep photos Pictures Public snap Templates Videos
```

- **file $(ls -d /usr/bin/* | grep zip)**
  - ✓ /usr/bin에서 zip이라는 이름이 포함된 파일의 파일 타입을 출력

리눅스 명령어(3)

Quoting

: Expansion을 통제하는 방식

```
joonhee@joonhee-laptop:~$ echo The total is $100.00
The total is 00.00
```

```
joonhee@joonhee-laptop:~$ echo white-space    is compressed
white-space is compressed
```

원치 않는 결과들

word-splitting

: 떨어진 단어는 별개 취급

Double Quotes

: $, ₩(backslash), `(back-quote)를 제외한 모든 기호 효과 무효화

```
joonhee@joonhee-laptop:~$ ls -l two words.txt
ls: cannot access 'two': No such file or directory
ls: cannot access 'words.txt': No such file or directory
joonhee@joonhee-laptop:~$ ls -l "two words.txt"
-rw-rw-r-- 1 joonhee joonhee 3  4월  7 01:34 'two words.txt'
```

```
joonhee@joonhee-laptop:~$ echo "$USER"_"$((1+1))_$(ls -d D*l*)"
joonhee_2_Downloads
```

괄호 안은 다른 영역 취급

```
joonhee@joonhee-laptop:~$ ls -d D*l*
Downloads
```

```
joonhee@joonhee-laptop:~$ ls -d "D*l*"
ls: cannot access 'D*l*': No such file or directory
```

```
joonhee@joonhee-laptop:~$ echo "whitespace          as you see"
whitespace          as you see
```

- **echo $(ls -l)**
- **echo “$(ls -l)” 차이 비교 (word-splitting 방지)**

리눅스 명령어(3)

Quoting

Single Quotes

: 모든 기호 효과 무효화

```
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
```

Escaping Characters

$, !, “, ‘, &, space, 기타 등등의 기호 앞에 ₩(backslash)써서 효과 무효화

```
joonhee@joonhee-laptop:~$ echo "Ontong-Daejeon for $USER remains: \$13.7"
Ontong-Daejeon for joonhee remains: $13.7
```

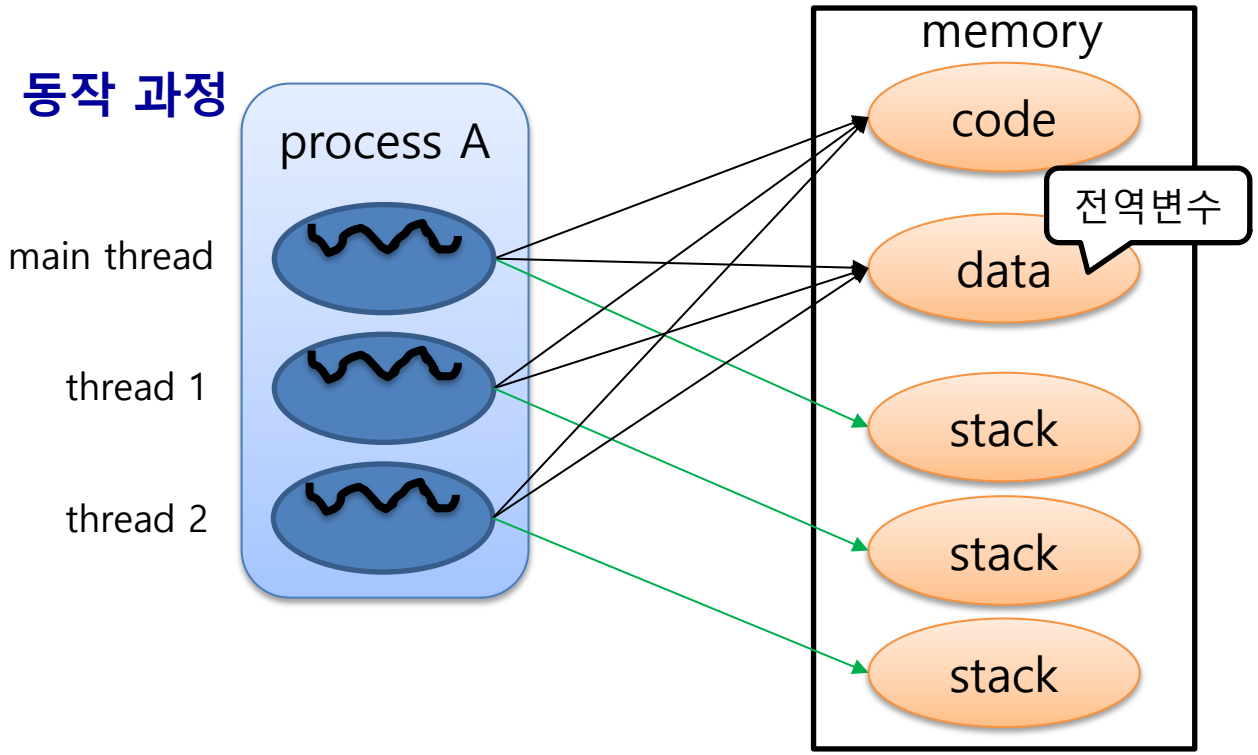
```
joonhee@joonhee-laptop:~$ echo "babo" > "bad&filename"
joonhee@joonhee-laptop:~$ ls -l bad\&filename
-rw-rw-r-- 1 joonhee joonhee 5  4월  7 01:58 'bad&filename'
joonhee@joonhee-laptop:~$ mv bad\&filename goodname
joonhee@joonhee-laptop:~$ ls -l goodname
-rw-rw-r-- 1 joonhee joonhee 5  4월  7 01:58 goodname
```

LIBRARY FOR THREADS

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

Library for Threads (pthread.h)

- **쓰레드란?**
  - **: 프로세스 내 하위 '실행 단위' (스케줄링 단위)**
    - **Linux의 thread는 kernel thread**
  - **주소 공간이나 열린 파일 등 여러 자원을 공유할 수 있음**
    - **프로세스에 비해 쓰레드 간 통신이 간편하고 생성시간도 짧음**
  - **동작 과정**



Library for Threads (pthread.h)

- **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

Library for Threads (pthread.h)

- 예제1
  - pthread_create, join

```
07:14 $ gcc thread_create.c -o a -Wall -pthread
```

```
07:00 $ ./a
10 20
done
```

assert:
만약 1로 컴파일할 경우의 결과

```
06:55 $ ./a
a: thread_create.c:21: main: Assertion `rc == 1' failed.
Aborted (core dumped)
```

```c
#include <assert.h>
#include <stdio.h>
#include <pthread.h>

typedef struct {
    int a;
    int b;
} myarg_t;

void *mythread(void *arg) {
    myarg_t *args = (myarg_t *) arg;
    printf("%d %d\n", args->a, args->b);
    return NULL;
}

int main(int argc, char *argv[]) {
    pthread_t p;
    myarg_t args = { 10, 20 };

    int rc = pthread_create(&p, NULL, mythread, &args);
    assert(rc == 0);
    pthread_join(p, NULL);
    printf("done\n");
    return 0;
}
thread_create.c" 25 lines --8%--
```

void *는 아무 포인터나 받을 수 있음

내부 변수 참조 사용하기 위해 형변환

리턴 필요없으면 NULL

뭘 써야 할지 모르겠는 포인터는 대부분 그냥 NULL을 쓰면 됨

Library for Threads (pthread.h)

- **예제2**
  - **pthread_create, join**

구조체를 사용하면 여러 인자를 넘길 수 있음

- 종료 결과를 저장할 변수 (여기서는 구조체)의 포인터

- 데이터 존속을 위해 heap에 할당 malloc/free

- main의 rvals에 저장됨

```
07:14 $ ./b
args 10 20
returned 1 2
```

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

typedef struct { int a; int b; } myarg_t;
typedef struct { int x; int y; } myret_t;

void *mythread(void *arg) {
    myarg_t *args = (myarg_t *) arg;
    printf("args %d %d\n", args->a, args->b);
    myret_t *rvals = malloc(sizeof(myret_t));
    assert(rvals != NULL);
    rvals->x = 1;
    rvals->y = 2;
    return (void *) rvals;
}

int main(int argc, char *argv[]) {
    pthread_t p;
    myret_t *rvals;
    myarg_t args = { 10, 20 };
    pthread_create(&p, NULL, mythread, &args);
    pthread_join(p, (void **) &rvals);
    printf("returned %d %d\n", rvals->x, rvals->y);
    free(rvals);
    return 0;
}
"thread_create_with_return_args.c" [Modified] 29 lines
```

Library for Threads (pthread.h)

- **예제3**
  - **pthread_create, join**

단순한 인자, 형변환
포인터를 일반 변수값처럼 사용

```
 1 #include <stdio.h>
 2 #include <pthread.h>
 3
 4 void *mythread(void *arg) {
 5     long long int value = (long long int) arg;
 6     printf("%lld\n", value);
 7     return (void *) (value + 1);
 8 }
 9
10 int main(int argc, char *argv[]) {
11     pthread_t p;
12     long long int rvalue;
13     pthread_create(&p, NULL, mythread, (void *) 100);
14     pthread_join(p, (void **) &rvalue);
15     printf("returned %lld\n", rvalue);
16     return 0;
17 }
18
thread_create_simple_args.c" [Modified] 18 lines --44%--
```

- 종료 결과를 저장할 변수의 포인터(주소)

- 주소를 일반 변수값처럼 사용하여서 복사 가능

```
07:22 $ ./c
100
returned 101
```

Library for Threads (pthread.h)

■ pthread.h : thread 관련 라이브러리

함수명	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
int pthread_mutex_trylock(pthread_mutex_t *mutex);	- lock 획득 - 누군가 lock을 갖고 있으면, block하지 않고 그냥 즉시 리턴

Library for Threads (pthread.h)

pthread.h : thread 관련 라이브러리

함수/변수명	설명
int pthread_mutex_init(pthread_mutex_t *restrict *mutex*, const pthread_mutexattr_t *restrict *attr*);	- pthread_mutex_t 변수(lock) 초기화 성공시 0, 실패시 에러 넘버 리턴 (이하 동일)
int pthread_mutex_destroy(pthread_mutex_t **mutex*);	- pthread_mutex_t 변수(lock) 사용중지
pthread_mutex_t *mutex* = PTHREAD_MUTEX_INITIALIZER;	- 초기화 상수 - pthread_mutex_init(&mutex, NULL)과 동일

Library for Threads (pthread.h)

예제4

pthread_mutex_t, lock, unlock

```
pthread_mutex_t lock;
pthread_mutex_lock(&lock);
x = x + 1; // or whatever your critical section is
pthread_mutex_unlock(&lock);
```

entry section

exit section

```
pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
int rc = pthread_mutex_init(&lock, NULL);
assert(rc == 0); // always check success!
```

1

2

1 또는 2 아무거나

과제

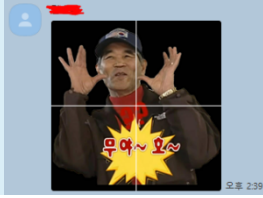
1. mutex

과제

설명은
다음 장

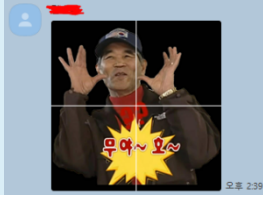
```
 1 #include <stdio.h>
 2 #include <pthread.h>
 3
 4 static volatile int counter = 0;
 5 static const int end = 100000000;
 6
 7 typedef struct { char *id; int val; } myarg;
 8
 9 void *mythread(void *arg)
10 {
11     myarg *ma = (myarg *) arg;
12     printf("%s[%u]: begin\n", ma->id, (unsigned) pthread_self());
13     for (int i = 0; i < end; i++) {
14         --counter;
15         ma->val++;
16     }
17     printf("%s[%u]: done\n", ma->id, (unsigned) pthread_self());
18     return (void *) ma;
19 }
20
21 int main()
22 {
23     printf("main[%u]: begin (counter = %d)\n", (unsigned) pthread_self(), counter);
24     pthread_t t1, t2;
25     myarg ma1 = {"A", end };
26     pthread_create(&t1, NULL, mythread, &ma1);
27     for (int i = 0; i < end; i++)
28         counter++;
29     pthread_join(t1, (void **) &ma1);
30     printf("main[%u]: done (counter = %d) (ma1.val = %d)\n",
31             (unsigned) pthread_self(), counter, ma1.val);
32     return 0;
33 }
"simple-race.c" [Modified] 34 lines --88%--                    30,45         Top
```

과제



- 1. 여러 차례 실행해보고 결과 확인 및 분석
  - 그러한 결과가 나온 이유에 대해서 설명
- 2. 29번 줄 pthread_join을 주석처리하면 어떻게 되는지 결과 확인 및 분석
  (수행 후 원상복귀)
- 3. 주어진 소스코드에 critical section이 있다. 어디 있는지 줄 번호를 모두 쓰고, 왜 그것들이 critical section인지 설명
- 4. 뮤텍스(lock) 변수를 이용하여 critical section이 mutual exclusive하게 실행되도록 코드를 작성하고, 실행 결과 확인. 이때 counter와 ma1.val의 최종 출력은 몇이어야 할까?
- 5. 4번이 완성된 경우, main thread에서
  쓰레드를 1개 더 추가하고 (t2; ma2 = {“B”, end }; create(ma2);)
  최종 출력 전에 join(ma2)을 추가한다고 가정하면,
  이때 counter의 최종 출력은 몇인가?
  (추가되는 코드는 모두 각각 종류가 같은 기존의 코드 직후에 추가된다고 가정)

과제



- 포함 내용
  - 소스코드
    - mutex_학번.c (4번 수행 후)
  - 보고서
    - 각 단계별 (1~5) 수행 결과 작성
    - 느낀 점 (쪽지시험, 수업, 과제 재미, 난이도 기타 등등)

- 제출기한 및 양식
  - ~4월 13일(화) 23:59
  - OS01_04_학번_이름.zip
    - src 폴더 이하 소스코드
    - 보고서 pdf 같은 이름

Q&A

Thank you!!!