운영체제

Pintos Project #1 - Pintos 환경 구축

손성훈 교수님

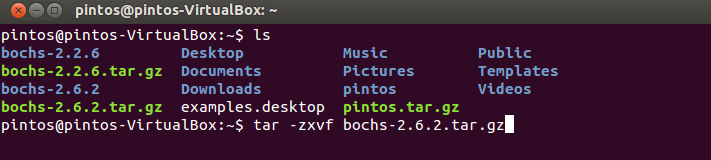
컴퓨터과학과

컴퓨터과학전공

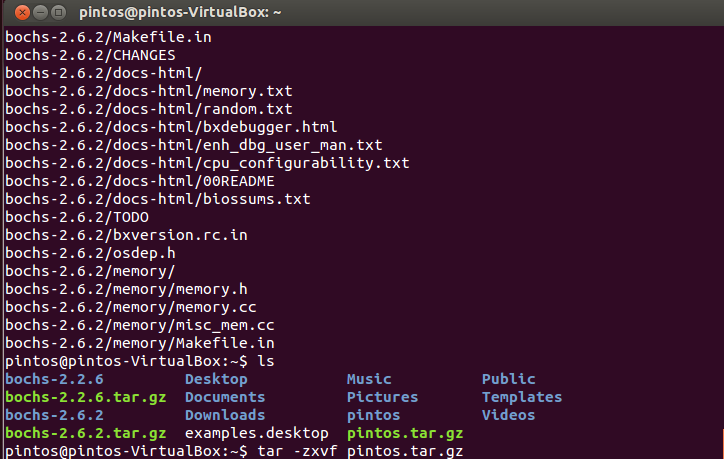
201311126

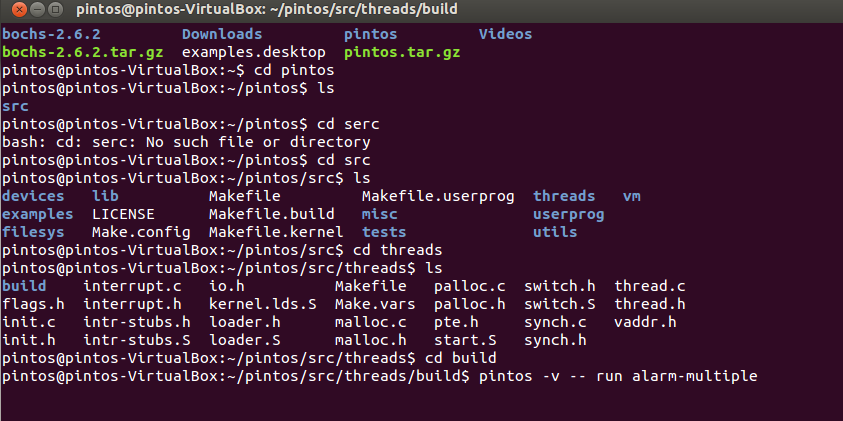
서우일

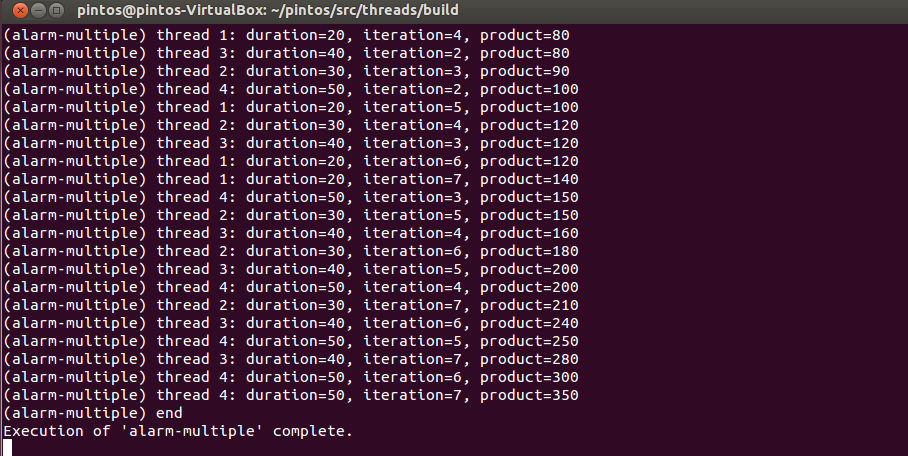
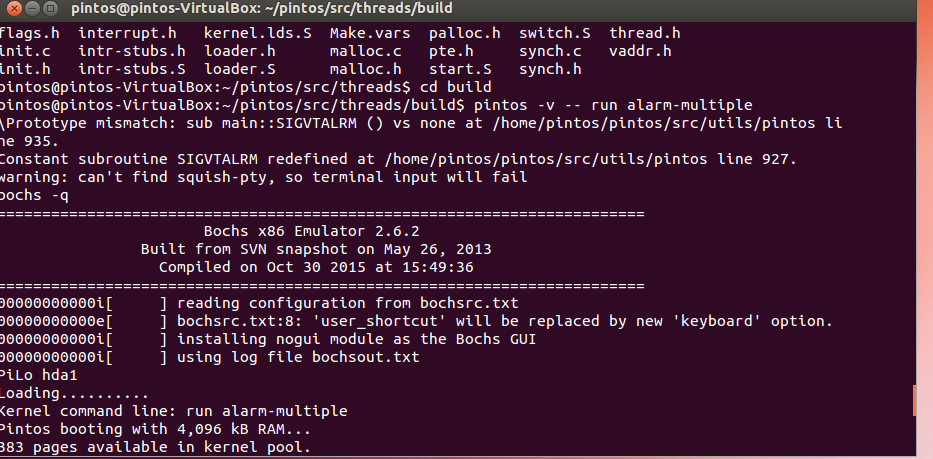
설치과정



ls로 현재 경로에 있는 파일들의 목록을 확인하고 tar.gz파일 압축을 풀기위해 tar -zxvf bochs-2.6.2.tar.gz를 하여 압출을 풀었습니다.

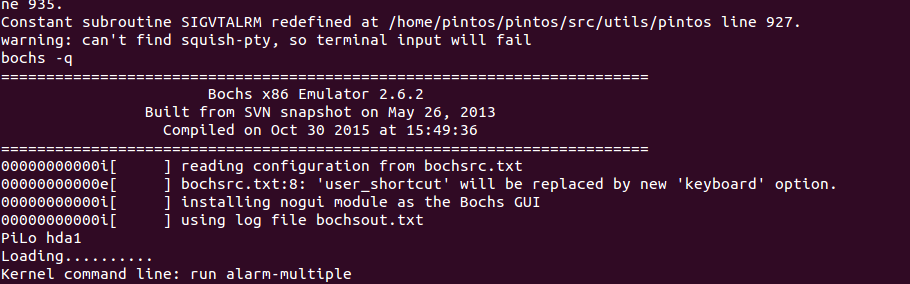


압축이 풀리고 나서 pintos.tar.gz의 압축을 해제하기위해 tar -zxvf pintos.tar.gz명령어를 사용했습니다.

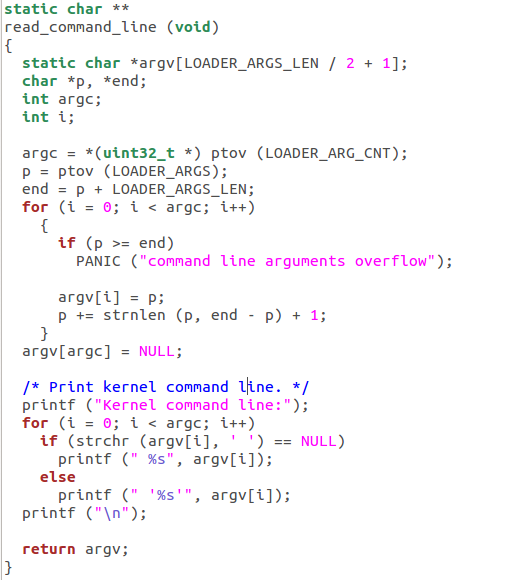
압축이 다 풀리고 나서 /pintos/src/threads/build 경로로 찾아가 pintos -v -- run alarm-multiple 명렁어를실행하였습니다.

--------------------------------------------------------------------------------------------------------------

명령어 실행과정 분석



Kernel command line: run alarm-multiple ->



커맨드 라인을 처리하는 함수로 보입니다.

argv를 포인터로 정의해서( static cha \*argv) argc = \*(uint32\_t \*)ptov (LOADER\_ARG\_CNT); (함수로 보이는데 어느 헤더파일에서 불러왔는지 잘 모르겠습니다) 로 argc에 값을 넣습니다

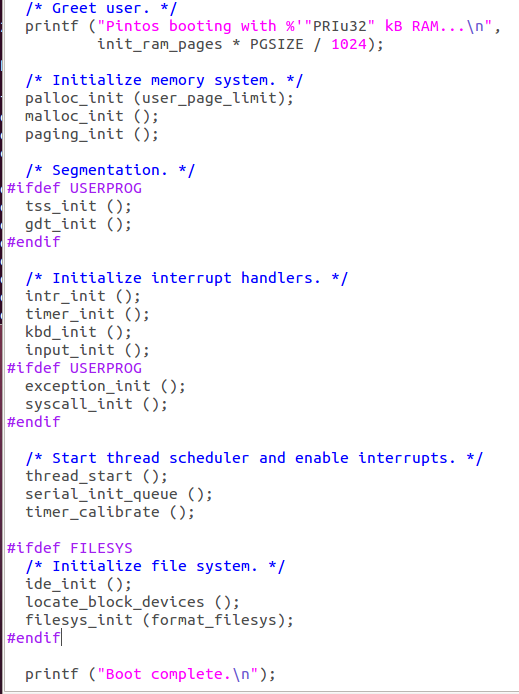
p가 end값보다 같거나 큰 경우 “command line arguments overflow”를 출력하고 argv[i]에 p를 입력합니다.

p에 strnlen에 p와 end - p를 파라미터로 같는 함수의 값에 +1을 하여 p에 값을 더합니다

Argv[i]배열에 각각 무슨 커맨드를 말해줄 지가 적혀있는데, strchr의 값이 0일 경우의 argv[i]의값을 출력하고, 또 다른 i에 대해서 맞는 문자열을 출력합니다. 함수가 끝나고 argv를 반환하고 함수를 종료합니다.



Pintos booting with 4,096 kB RAM... ->

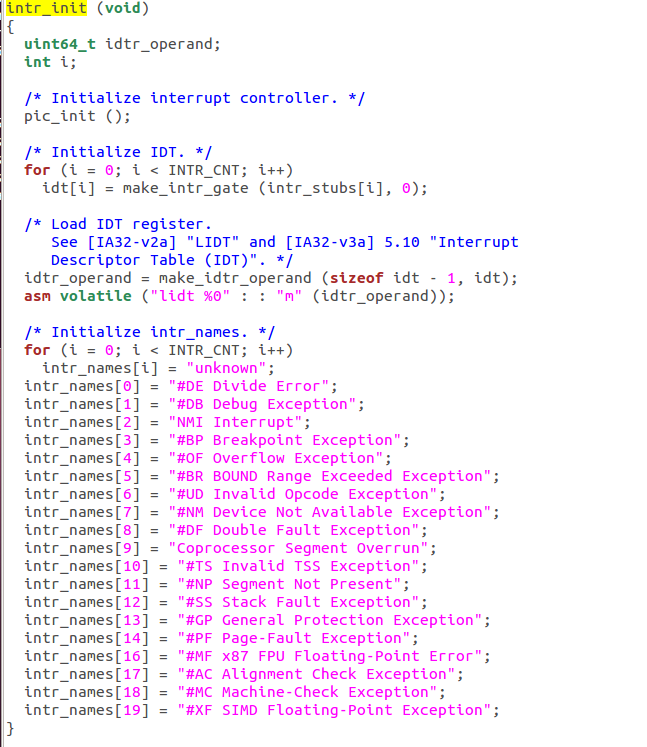


PRIu32의 값인 init\_ram\_pages \*PGSIZE의 값을 1024로 나누어서 나온 결과를 문자열로 출력합니다.

메모리를 초기화합니다. Palloc.c 파일의 palloc\_\_init함수에(user\_page\_limit)을 인자로 받아서 함수를 실행하고, malloc\_init()과(malloc.c참조) paging\_init()함수를 실행하여 초기화합니다.

/\* Initialize interrupt handlers \*/

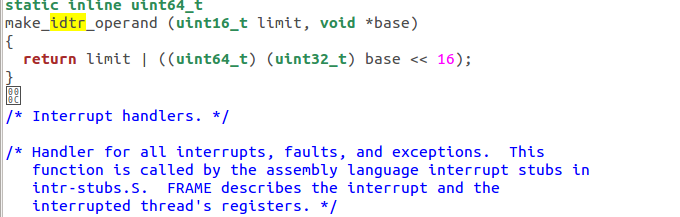
Intr\_init(); -> 인터럽트를 초기화합니다.

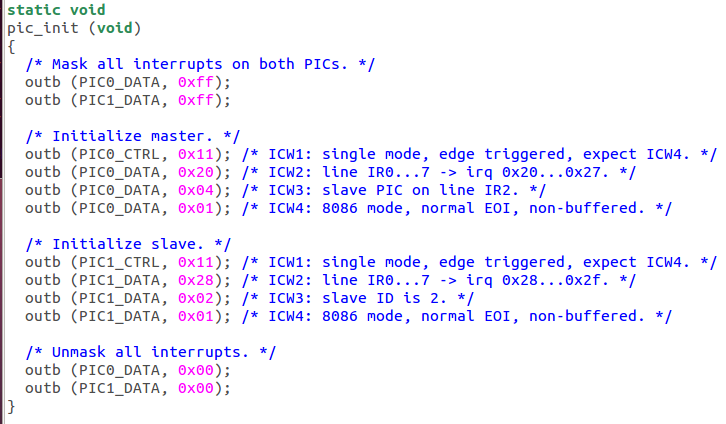


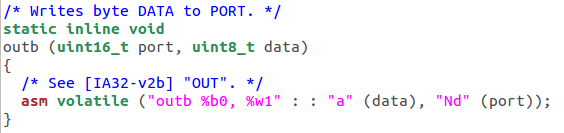
Interrupt.c 파일에 있는 intr\_init()함수를 호출하여 초기화합니다.

인터럽트 컨트롤러를 초기화한 후에 총 20개의 인터럽트를 초기화 합니다.

Idtr\_operand : Interrupt Descriptor Table의 시작주소를 가리키는 Register를 위한 값을 만들기위한 공간인데, 아래의 make\_idtr\_operand함수에 idt-1사이즈만큼 과 idt값을 넘겨서 리턴한 값을 idtr\_operand에 넣습니다.



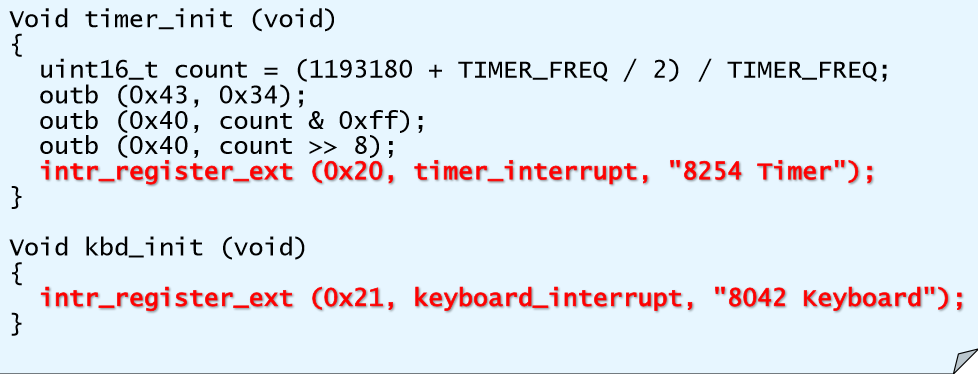
Pic\_init() : 인터럽트를 입력받은 대로 목록에서 골라오는 역할을 하는 인터럽트 컨트롤러를 초기화 합니다.



Outb 명령어는 쓰기를 위한 명령어로 각각 앞의 주소에서 오른쪽 주소에 쓰기 합니다

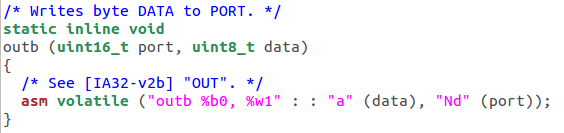
Timer\_init(); -> 타이머를 초기화하는 함수를 실행하여 초기화 합니다.

\*함수가 src/thread/ 경로 안 파일에는 보이지 않아서 수업자료를 활용하였습니다

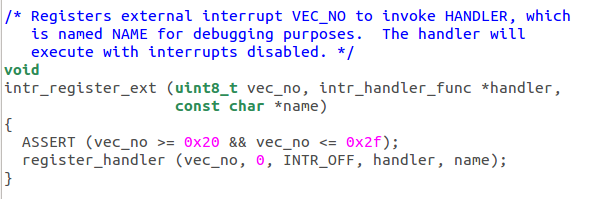


.count : 자료형 uint16\_t로 타이머 주기값을 이용하여 계산후 저장합니다

Outb (0x43, 0x34)-> 0x43주소에 있는 값을 0x34주소에 쓰기 합니다

...

Intr\_register\_ext (0x20, timer\_interrupt, “8254 Timer”); ->



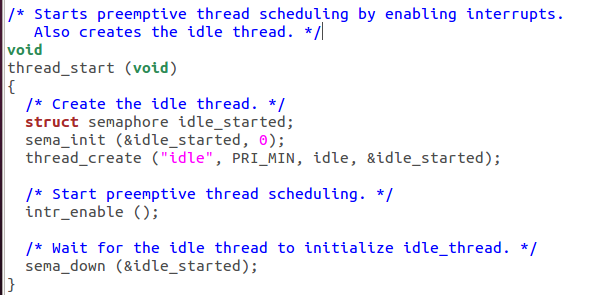
인수로 받은 vec\_no(현재 처리하고있는 인터럽트 값)값이 ASSERT 조건에 확실하게 성립하면(0x20값보다 크거나 같고, 0x2f보다 작거나같아야) register\_handler 함수를 실행합니다.

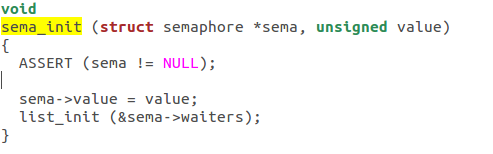
Kbd\_init(); -> 키보드 인터럽트를 초기화합니다 -> 위의 intr\_register\_ext함수로 타이머와 키보드 인터럽트 초기화를 처리합니다.

Input\_init(); -> 입력을 초기화합니다.

/\* start thread scheduler and enable interrupts. \*/

thread\_start(); ->





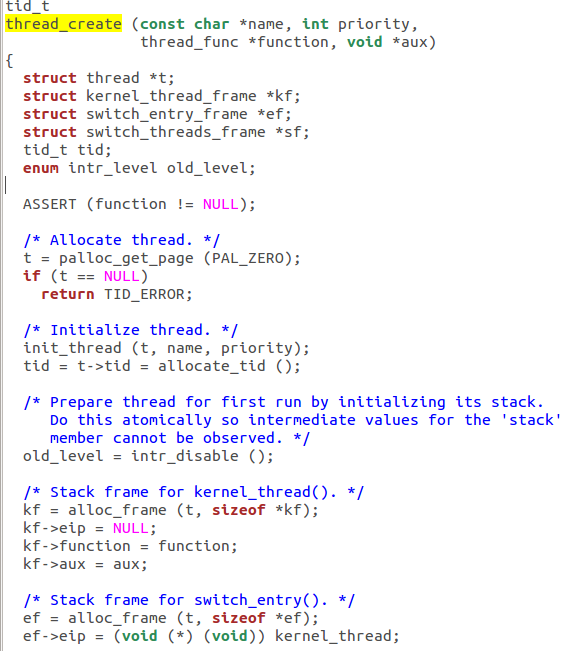
쓰레드를 시작합니다

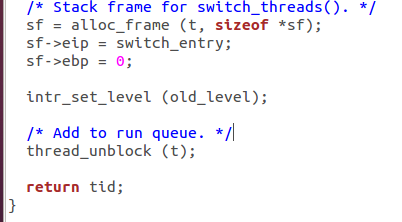
세마포어 형식의 구조체 idle\_started를 만듭니다

Sema\_init 함수에 idle\_started 주소값을 매개변수로 하여 sema가 NULL이 아닌 경우에

세마의 밸류값에 밸류를 넣고, list\_init함수에 sema의 waiter를 매개변수로 하여 실행합니다.

Thread\_create (”idle”,PRI\_MIN, idle, &idle\_started); ->





struct thread \*t -> t 라는 thread 구조체를 만듭니다

struct kernel\_thread\_frame \*kf -> kf라는 kernel\_thread\_frame 구조체를 만듭니다

struct switch\_entry\_frame \*ef -> ef라는 switch\_entry\_frame 구조체를 만듭니다

struct kernel\_thread\_frame \*sf -> sf라는 kernel\_thread\_frame 구조체를 만듭니다

tid\_t tid -> tid\_t 자료형 tid를 선언합니다

enum intr\_level old\_level -> 열거형 선언으로 intr\_level 과 old\_level을 선언합니다.

ASSERT (function != NULL); -> function 의 값이 NULL이 아니어야 합니다.

t = palloc\_get\_page (PAL\_ZERO); -> palloc으로 할당한 메모리공간을 t에 저장합니다

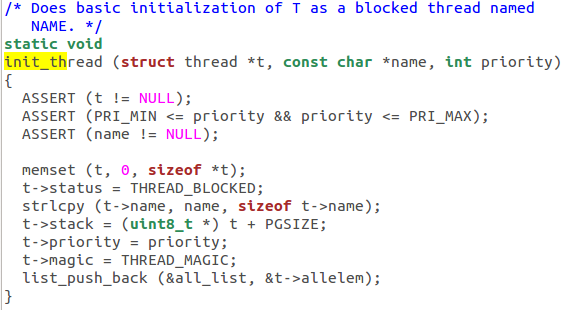
If ( t==NULL)

Return TID\_ERROR;

t에 값이 저장되지 않은경우에는 TID\_ERROR를 반환합니다.

-------------------------------------------------------------------------------------------------------------

init\_thread (t, name, priority); -> 매개변수 t, name, priorty를 받아서 Init\_thread함수를 실행합니다.



ASSERT (t != NULL);

ASSERT (PRI\_MIN <= priority && priority <= PRI\_MAX);

ASSERT (name != NULL);

t가 NULL이 아니고 PRI\_MIN이 priority보다 작거나같거고 priority가 PRI\_MAX보다 작거나 같고, 또한 name값이 NULL이 아니어야 합니다.

memset (t, 0, sizeof \*t); -> t, 0 , t의 크기를 매개변수로 memset함수를 실행합니다.

t->status = THREAD\_BLOCKED; -> t의 status값에 THREAD\_BLOCKED값을 저장합니다

strcpy (t->name, name, sizeof t->name); -> t의 name 문자열을 name에 복사합니다.

t->stack = (uint8\_t \*) t+ PGSIZE; -> t의 stack값에 uint8\_t자료형 t + PGSIZE값을 저장합니다

t->priority = priority; -> t의 priority에 priority값을 저장합니다

t->magic=THREAD\_MAGIC -> t의 magic값에 THREAD\_MAGIC값을 저장합니다;

list\_push\_back(&all\_list, &t->allelem); -> list\_push\_back함수를 all\_list와 t->allelem의 주소값을 매개변수로 받아서 실행합니다.

-------------------------------------------------------------------------------------------------------------

tid = t->tid = allocate\_tid(); -> tid와 t->tid값에 allocate\_tid()함수 return값을 저장합니다.

old\_level = intr\_disable(); -> 열거형으로 선언했던 변수 old\_level에 intr\_disable()함수가 리턴한 값을 저장합니다.

kf = alloc\_frame (t, sizeof \*kf); -> kf에 alloc\_frame함수의 return값을 저장합니다

kf->eip = NULL; -> kf구조체의 eip값에 NULL값을 저장합니다

kf->function = function; -> kf구조체의 function값에 function값을 저장합니다

kf->aux = aux; -> kf 구조체의 aux값에 aux값을 저장합니다

ef = alloc\_frame (t, sizeof \*ef) -> ef에 alloc\_frame함수의 return값을 저장합니다

ef->eip = (void (\*) (void)) kernel\_thread; ef의 eip값에 kernel\_thread값을 저장합니다

sf = alloc\_frame(t, sizeof \*sf); -> sf에 공간을 할당합니다

sf->eip = switch\_entry; sf구조체의 eip값에 switch\_entry값을 저장합니다.

sf->ebp = 0; sf구조체의 ebp값에 0을 저장합니다

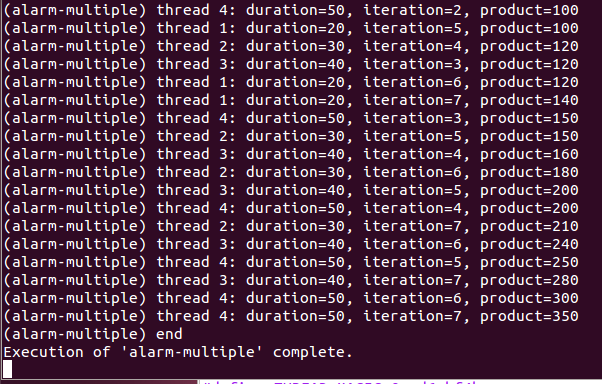
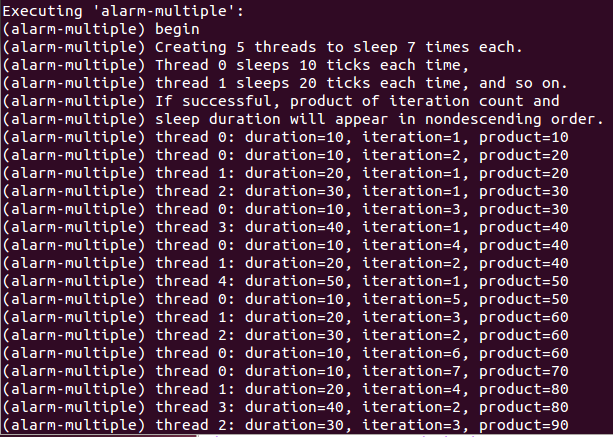
intr\_set\_level (old\_level); old\_level값을 매개변수로 받은 intr\_set\_level을 실행합니다.

thread\_unblock (t); run queue에 들어가기 위해 쓰레드를 unblock하기위한 함수를 호출합니다.

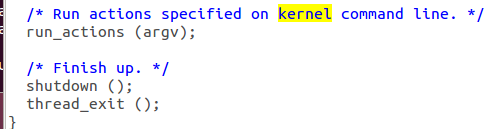
return tid; tid값을 반환하고 함수를 종료합니다

serial\_init\_queue (); -> 레디큐 진입 순서를 카운트시작합니다

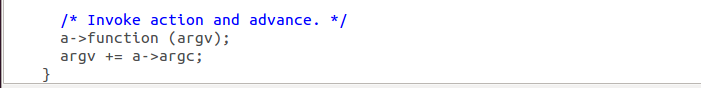
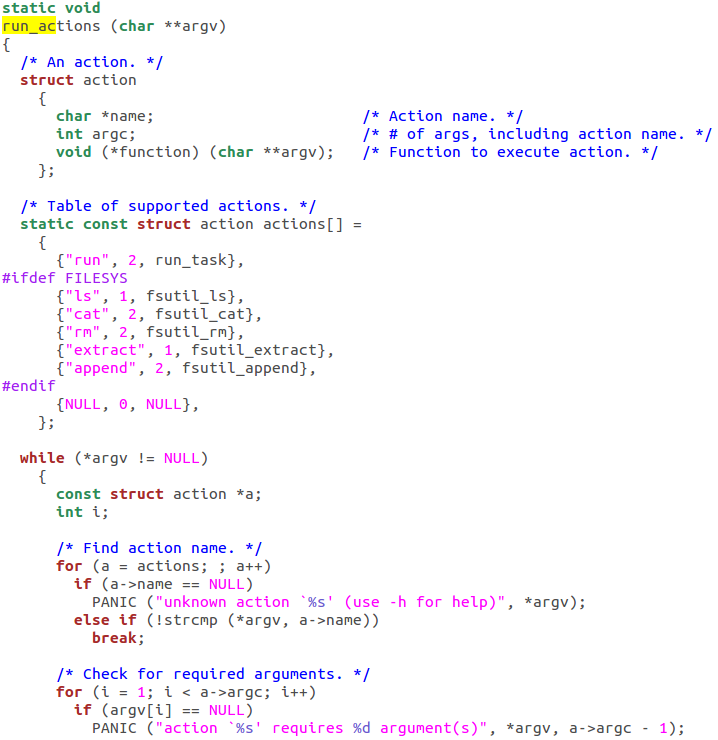
timer\_calibrate () -> 시간 측정을 시작하는 함수를 호출합니다



Executing ‘alarm-multiple’ ->



run\_actions(argv); -> argv를 매개변수로 하여 run\_actions 함수를 실행합니다



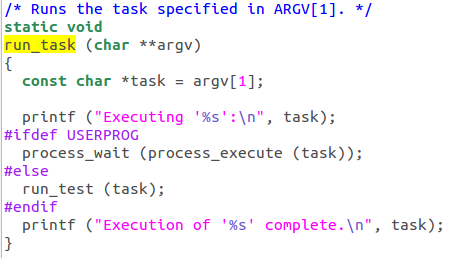
struct action {

action 구조체 변수들을 선언합니다

}

action타입 action[]배열을 만들고,run\_task 함수를 실행합니다.

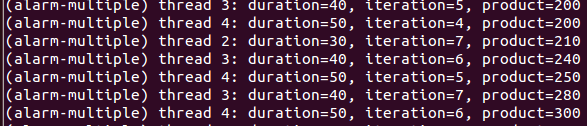
-------------------------------------------------------------------------------------------------------------



포인터변수task에 argv[1]의값을 저장합니다.

Executing task를 출력합니다.

이 함수를 통해서



이와 같은 출력을 만들어냅니다.

경우에따라 USERPROG이 1인경우 프로세스가 wait상태가 되고, 다른경우는 run\_test를 계속 실행합니다.

마지막인경우 Execution of task complete를 출력합니다.



--------------------------------------------------------------------------------------------------------------

argv의 값이 NULL이 아닌 경우 while구문을 실행합니다. action name을 찾고 a의 name값이 NULL인경우 unknown action을 출력합니다. 아닌경우는 strcmp를 실행하여 문자열의 바이트를 비교합니다

아래의 for 구문에서는

Argv[i]의 값이 NULL인 경우 패닉함수를 출력합니다.

a->function (argv); -> a의 function함수를 argv를 받아서 실행합니다

argv += a->argc; -> argv에 a의 argc값을 더합니다

프로그램 실행 경로 요약

1. 커맨드라인에서 명령어를 확인합니다

2. 메인함수에서 핀토스 부팅을 시작합니다

3. 부팅과정에서 메모리들과 쓰레드, 인터럽트가 초기화가 이루어집니다.

4. 타이머와 인터럽트핸들러가 초기화되고 인터럽트 디스크립터도 초기화됩니다.

5. 쓰레드를 시작하고 레디큐 순서를 카운트하기 시작하고 타이머가 시작됩니다

6. 부팅이 완료되고 alarm-multiple 명령어가 실행됩니다

7. 쓰레드가 나누어서 명령어를 처리합니다.

8. 마지막 명령어를 처리하고 프로그램을 종료합니다.

기여도 평가

201311126 서우일 4 Ticket

------------------------------------------------------------------------------

TAGS 사용에 대해 인지를 뒤늦게 해서... 서술식으로 적었습니다

스크린샷과 바로 이어서 글을 썼습니다.. 다음부턴 꼭 태그를 사용해서 적도록 하겠습니다..