운영체제의 기초 Project 2

전기정보공학부

2017-17088 박찬정

1. os\_context\_t 구조체

해당 구조체에는 edi, esi, ebx, edx, ecx, eax, \_eflags, eip 로 총 8개의 멤버 변수를 정의하였다. 모두 int32u\_t 타입이다.

2. \_os\_save\_context 함수

차례로 $resume\_eip, \_eflags, eax, ecx, edx, ebx, esi, edi 값을 stack에 push한다. 그리고 당시의 stack pointer의 위치를 eax에 저장한다 (리턴 값 준비). 이후 base pointer 아래에 위치하는 old ebp, old eip 값을 stack에 push하여 해당 값을 보호하고, base pointer를 stack top으로 옮긴다.

3. \_os\_restore\_context 함수

해당 함수의 내용은 모두 인라인 어셈블리로 구현하였다. 우선 함수의 input으로 들어온 addr\_t 타입의 sp 값을 인라인 어셈블리의 인풋으로 제공하였으며, 해당 값으로 stack pointer를 이동시키며 시작한다. eip를 제외한 7개의 context 값을 차례로 pop한 뒤, esp 값에서 4만큼 다음에 있는 값을 base pointer에 저장한다. 이는 old EBP 값에 해당한다. 이후 ret 명령어를 수행하면 원래의 context를 모두 회복한다.

4. \_os\_create\_context 함수

처음 task가 만들어질 때, 실행 중간에 멈춘 context처럼 만들어주는 함수이다. 그래야 나중에 schedule시에 같은 방법으로 context를 restore하고서 정상적으로 수행할 수 있다.

우선 stack의 가장 high address를 찾는다. 이것이 stack에서는 가장 밑바닥이 된다. 그리고 차근차근 arg, return address (현재로서는 NULL), eip (현재로서는 entry), \_eflags (현재로서는 1) 값을 stack의 밑바닥부터 저장한다. 이후 6개의 레지스터 값을 stack에 저장한다. 저장이 끝나면 stack top에 해당하는 주소를 반환한다. 주어진 task의 stack 안에서 연산하는 것이 아니고 그 stack의 내용을 꾸며주는 것이므로 어셈블리가 아닌 c 코드로 작성한다.

5. eos\_tcb\_t 구조체

Task에 대한 정보를 저장하는 구조체이므로, state, stack pointer, stack base, stack size, entry, argument for entry, node for queueing 의 내용을 저장한다.

6. eos\_create\_task 함수

우선 위에서 정의한 \_os\_create\_context 함수를 사용하여 주어진 stack에 context를 꾸며놓는다.

다음으로 주어진 tcb에 task의 Stack에 대한, entry와 arg에 대한 정보를 기록한다.

이제 queueing\_node를 생성한다. malloc으로 생성하고 데이터를 저장한다. 이후 tcb에 queueing\_node를 저장한다.

이제 context와 tcb에 대한 처리가 끝났으므로 해당 task를 queue에 push한다.

7. eos\_schedule 함수

우선 현재의 task가 있는지 확인한다. 있다면, 당시의 context를 저장하고 queue의 맨 뒤로 보낸다.

이제 가장 높은 priority queue에서 가장 앞에 있는 task를 가져온다. 해당 task를 \_os\_current\_task의 값으로 넣어 앞으로 수행하도록 한다. 그리고 queue에서 task를 제거하고, context를 복원한다.