시스템프로그래밍 실습

과제 4 kernel lab

2015-16828 김민규

1. 서론

ptree 부분은 process id를 받아, init부터 시작하여 그 id를 가진 process까지의 process들을 출력하는 과제였습니다. paddr부분은 virtual address를 받았을 때, 그 address가 참조하는 실제 physical address를 반환하는 과제였습니다.

2. 본론

2.1. ptree 구현

2.1.1. write_pid_to_input 함수

input을 처리하고 작업을 수행하기 위한 함수입니다.
먼저, sscanf 함수를 이용해 user_buffer에서 pid를 받아옵니다. pid_task함수를 이용해 pid에 해당하는 task를 받아옵니다. 각 노드가 char 배열을 가지는 링크드 리스트를 선언한 뒤, 첫 번째 노드의 배에 task의 정보를 sprintf함수를 이용해 "task_name (pid)\n" 형태로 담습니다. 이 때, spirntf함수는 그 길이를 반환하므로, 총 길이를 담을 변수 total_len에 그 길이를 더합니다. 그 다음 이 task의 부모 process의 정보를 링크드 리스트의 다음 노드에 담습니다. 이 때 list_add 함수를 이용해 다음 항목이 이전 항목보다 앞에 오게끔 합니다. 이는 process 정보는 역순으로 init이 나올 때까지 거슬러 올라가지만, 실제 출력은 init부터 시작하여 내려가야 하기 때문입니다. 이 작업을 더 이상 부모 process가 없을 때까지(즉, init에 도달할 때까지) 반복합니다.

2.1.2. read_output 함수

위에서 전역변수 output 에 담긴 정보를 user_buffer로 옮기기 위한 함수입니다. temp 라는 char배열을 위에서 구한 total_len길이만큼 선언해, list_for_each로 리스트를 순회하며 각 문자열을 strcat함수를 이용해 temp배열에 더합니다. simple_read_from_buffer라는 함수를 이용하여 user_buffer에 temp의 정보를 담습니다. 그 다음, list_for_each_safe함수로 다시 리스트를 순회하며 list_del함수를 이용해 각 노드를 삭제하고, kfree함수로 free시켜 줍니다.

2.1.3. __init dbfs_module_init 함수

먼저, debugs_creat_dir이라는 함수로 kernel의 현재 디렉토리에 ptree라는 하위 디렉토리를 만들고, 그것을 dir(전역변수)에 담습니다. 그 다음 debugfs_create_file이라는 함수로 위의 dir에 input이라는 file을 만듭니다. 이를 직접 쓰지는 않지만, 이 파일을 만드는 과정에서 write_pid_to_input함수를 실행시켜 output배열에 정보를 담는 역할을 합니다. 다음으로 debugfs_create_file함수를 다시 실행시켜, 이번에는 dir라는 위치에 ptree라는 파일을 만듭니다. ptree를 만드는 과정에서 위에서 만든 read_output함수를 실행시키게 되고, ptree에 output의 결과물이 담기게 됩니다.

2.1.4. __exit dbfs_module_init 함수

debugfs_remove_recursive(dir)을 실행시켜, dir상에 만들어진 파일들을 삭제합니다.

2.2. paddr 구현

2.1.1. read_output 함수

input을 처리하고 작업을 수행하기 위한 함수입니다.
먼저, copy_from_user라는 함수를 이용해 copied라는 char 배열에 user_buffer의 메모리를 복사해 옵니다. user_buffer에는 app.c에서 넘겨준 pckt structure가 담겨 있습니다. pckt structure에는 pid,vaddr,paddr(app.c에서 paddr에 해당하는 부분은 0으로 넘겨줍니다) 세 개의 변수가 담겨 있으며 struct는 가장 큰 변수 타입에 맞춰 메모리를 할당하므로 각 8바이트씩을 차지하게 됩니다. 따라서 user_buffer에는 24바이트가 연속적으로 담기게 됩니다. 이를 가지고 pid, vaddr 정보를 분리해 냅니다.
먼저 copied배열에서 pid_t만큼의 사이즈를 memcpy하여 input_pid에 담습니다. 그 다음, copied+8 주소에서 8바이트만큼 memcpy하여 vaddr에 담으면 분리가 완료됩니다.
다음으로, pid_task함수로 input_pid에 해당하는 task를 받아온 뒤, pgd_offset, p4d_offset, pud_offset, pmd_offset, pte_offset, pte_pfn 함수들로 page walk 를 하여 pfn을 추출해 냅니다. 5 level page table에서는 pfn뒤에 12비트의 offset을 붙이므로, pfn<<12 + vaddr%4096으로 paddr을 계산합니다. 그 다음, copied+16의 주소에 paddr의 내용을 memcpy로 복사합니다. 이제 copied 배열은 pid, vaddr, paddr 이 담기게 되고, 이를 simple_read_from_buffer함수로 user_buffer로 옮겨줍니다.

2.1.2. __init dbfs_module_init 함수

먼저, debugs_creat_dir이라는 함수로 kernel의 현재 디렉토리에 paddr라는 하위 디렉토리를 만들고, 그것을 dir(전역변수)에 담습니다. 그 다음, debugfs_create_file로 output이라는 파일을 그 dir에 만듭니다. 이 때, read_output함수를 실행시켜 작업을 수행하게 됩니다.

2.1.3. __exit dbfs_module_init 함수

먼저, debugs_creat_dir이라는 함수로 kernel의 현재 디렉토리에 paddr라는 하위 디렉토리를 만들고, 그것을 dir(전역변수)에 담습니다. 그 다음, debugfs_create_file로 output이라는 파일을 그 dir에 만듭니다. 이 때, read_output함수를 실행시켜 작업을 수행하게 됩니다.

3. 결론과 의견

리눅스 커널에 관한 프로그래밍은 처음 다뤄보기에, 굉장히 당혹스러웠습니다. 평소에 user space에서 별 생각 없이 쓰던 C 문법들이 커널에서는 쓸 수 없는 것들이 있었고, 커널에 관한 사전 지식이 풍부하지 못했기 때문이라고 생각합니다.