**Pintos Project 1: User Program (1)**

담당 교수 : 김영재

조 / 조원 : 20211558 윤준서

개발 기간 : 9.17 ~. 10.1 (15일)

1. **개발 목표**

* **해당 프로젝트에서 구현할 내용을 간략히 서술.**

• 프로그램의 실행 내용을 arguments로써 user stack에서 parsing 후 esp에 push한다.

• syscall handler 함수를 구현한다. 이때 process 관리를 위해 synchronization을 사용한다.

• fibonacci와 max\_of\_four\_int를 새로운 syscall로 정의하고 pintos가 해당 syscall를 처리할 수 있도록 한다.

1. **개발 범위 및 내용**
   1. **개발 범위**

* **아래 항목을 구현했을 때의 결과를 간략히 서술**

1. Argument Passing

esp가 word 단위를 벗어나지 않으며 올바르게 arguments들을 user stack으로부터 push한다. x86 규약에 따라 word alignment(4바이트 정렬) 를 만족시키고, 최종 esp가 user 영역(0xC0000000 미만) 에 있도록 보장한다.

1. User Memory Access

0xc00000000의 밑인 user 영역에서 stack push가 그 위인 kernel 영역을 침범하지 않고 진행된다. 잘못된 포인터/페이지 미매핑/널 포인터/kernel 영역 접근 시 프로세스를 종료한다.

1. System Calls

system call handler 함수 구현을 통해 여러 system call(halt, exit, read, write 등)을 호출한다. 추가로 등록한 fibonacci, max\_of\_four\_int 들을 system call에 등록한 후 파일 실행을 통해 값을 반환 시킬 수 있다.

* 1. **개발 내용**
* **아래 항목의 내용만 서술 (기타 내용은 서술하지 않아도 됨.)**
* Argument Passing
  + 커널 내 스택에 argument를 쌓는 과정 설명
* 프로그램을 실행하면 main(int argc, char \*argv[]) 의 인자들이 스택에 역순으로push된다. 이때 문자열은 메모리에 저장되고, 그 주소 값들이 스택에 저장된다.
* User Memory Access
  + Pintos 상에서의 invalid memory access 개념을 간략히 설명
* user program이 접근할 수 없는 메모리 영역(kernel)에 접근하려는 상황이다.
  + Invalid memory access를 어떻게 막을 것인지 설명
* esp 주소가 유효한지 확인하거나, system call을 통해 유효성을 검사할 수 있다.
  + 시스템 콜의 필요성에 대한 간략한 설명
* user program이 직접 하드웨어나 kernel 자원을 사용할 수 없으므로, kernel을 통해 요청하는 인터페이스가 필요하다. 이때 사용할 수 있는 API가 system call이다.
  + 이번 프로젝트에서 개발할 시스템 콜에 대한 간략한 설명 (하나의 시스템 콜 당 최대 3문장으로 간략히 설명. 3문장을 넘길 정도로 길게 작성하지 말 것)
* halt : pintos를 종료한다.
* exit : 현재 process를 종료하고 parent process에 종료 상태를 저장한다.
* exec : 새로운 process를 실행한다.
* wait : child process가 종료될 때까지 대기한다.
* read / write : 파일에서 데이터를 읽거나 쓴다.
* fibonacci : argument번째 피보나치 수를 반환한다.
* max of four int : 4개의 int arguments 중 최댓값을 반환한다.
  + 유저 레벨에서 시스템 콜 API를 호출한 이후 커널을 거쳐 다시 유저 레벨로 돌아올 때까지 각 요소를 설명
* user program이 library function을 통해 system call API를 호출한다. 그리고 kernel의 system call handler 함수가 호출되어 system call의 번호와 argument를 확인한다. 그 다음 system call service routine이 실행되고 그 결과 값을 레지스터에 저장한 뒤 iret을 통해 user mode로 복귀한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* **II. A.의 개발 범위를 포함하여 구현 내용에 대한 일정 작성**

• 9.17 ~ 9.20 : Argument Passing

• 9.21 ~ 9.24 : User Memory Access

• 9.25 ~ 9.28 : System Calls

• 9.28 ~ 10.1 : Additional System Calls

* 1. **개발 방법**
* **II. B.의 개발 내용을 구현하기 위해 어느 소스코드에 어떤 요소를 추가 또는 수정할 것인지 설명. (함수, 구조체 등의 구현이나 수정을 서술)**

• examples/additional.c : fibonacci, max\_of\_four\_int 실행용 파일 추가. argument가 2개면 fibonacci만, 5개면 fibonacci와 max\_of\_four\_int 모두 출력하도록 코드 작성.

• examples/Makefile : additional system call을 디버깅하기 위해 additional.o를 생성하도록 코드 추가.

• lib/syscall-nr.h : fibonacci와 max\_of\_four\_int용 system call을 추가로 등록한다.

• lib/user/syscall.c : additional.c에서 실행할 함수를 정의하고, 해당 함수가 호출할 system call을 연결한다. 이때 max\_of\_four\_int는 argument가 5개이므로 전용 syscall을 정의한다.

• threads/thread.h : thread 구조체에 child thread, child thread list, thread status, semaphore를 추가한다.

• threads/thread.c : thread.h에서 추가한 요소들을 추가 init 한다.

• userprog/process.c : argument parsing과 user stack allocation을 구현한다.

• userprog/syscall.c : kernel API를 작동시키기 위해 system call handler 함수를 구현한다.

1. **연구 결과**
   1. **Flow Chart**

* **II. B. 개발 내용에 대한 Flow Chart를 작성**

1. Argument Passing

텍스트, 도표, 스크린샷, 평면도이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. User Memory Access

텍스트, 도표, 평면도, 기술 도면이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. System Calls

도표, 평면도, 스케치, 그림이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

* 1. **제작 내용**
* **II. B. 개발 내용의 실질적인 구현에 대해 코드 관점에서 작성.**
* **구현에 있어 Pintos에 내장된 라이브러리나 자체 제작한 함수를 사용한 경우 이에 대해서도 설명.**
* **개발상 발생한 문제나 이슈가 있으면 이를 간략히 설명하고 해결책에 대해 설명.**

1. Argument Passing

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

우선 file name을 공백 단위로 쪼개고 각 word(argument)를 argv에 저장한다. 그리고 argument의 개수를 argc에 저장한다. 이를 esp(user 스택)에 역순으로 push하고, 각 스택의 주소를 argv\_ptr에 저장한다. x86 환경이므로 esp를 4byte 단위로 align을 한 뒤, 스택에 argv\_ptr에 저장했던 argv의 원소가 담긴 스택의 주소 자체를 다시 스택에 역순으로 push한다. 그리고 argv[]의 시작 주소, argc 값, return address를 스택에 순서대로 push한다. 최종적으로 스택 구조는 {argc, argv[0], argv[1], ...} 이 된다.

1. User Memory Access

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

userprog/syscall.c : arugment의 주소가 유효하거나 kernel 영역에 있는지, 또는 virtual adress가 실제 physical memory에 fetch 되어 있는지 검사한다. 만약 유효하지 않거나 kernel 영역 내에 있다면, 또는 physical memory에 존재하지 않다면 잘못된 포인터로 판단하고 kernel panic을 막기 위해 프로세스를 종료한다.

1. System Calls

* **이번 프로젝트에서 개발한 시스템 콜을 구현 관점에서 상세히 서술.**

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

userprog/syscall.c : esp가 유효한 위치(user program 영역)에 있는지 검사한 후, 전달받은 system call number에 따라 system call을 확인한다. system call handler 함수마다 각각 필요로 하는 argument 개수가 다르기 때문에, system call의 종류에 따라 esp 위치를 기준으로 스택에서 argument를 가져온다. 즉 argument parsing 처리라고 볼 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

이때 사용한 함수 get\_user\_int()의 코드는 다음과 같다. user address로부터 32bit 정수를 읽어 반환한다. 즉 kernel로 전달한다.

텍스트, 스크린샷이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

위에서 esp가 유효한 주소에 있는지 먼저 검사를 시행한 후, esp에 담긴 system call 번호를 get\_user\_int()를 통해 받았다. 이후 번호에 알맞은 system call을 처리하도록 구현했다. 이때 얻은 arg0 ~ arg3을 각 system call handler 함수에 필요한 argument type으로 변형한 뒤 system call handler 함수를 실행하도록 했다.

1. Additional System calls

* **새로운 시스템 콜(fibonacci, max\_of\_four\_int)을 구현하기 위해 수정하거나 작성한 코드에 대해 서술**

텍스트, 폰트, 스크린샷이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

lib/syscall-nr.h : 추가할 system call을 번호로써 해당 파일에 추가 등록한다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

lib/user/syscall.c : user program에서 system call을 호출하는 wrapper에서 argument가 5개인 경우를 추가했다(for max\_of\_four\_int). 스택에 역순으로 push를 하도록 했다. 그리고 additional system call을 호출하는 함수를 구현했다. 해당 argument들을 wrapper를 통해 kernel로 전달해서 처리하도록 한다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

examples/additional.c : additional system call을 pintos에서 직접 실행할 수 있도록 새로운 C 파일을 생성한다. file name과 함께 받는 argument들은 char type이므로 atoi()를 통해 int로 변환한다. 그리고 lib/user/syscall.c에서 구현한 system call 호출 함수를 arguments와 함께 실행한다. 이때 해당 디렉토리의 Makefile에도 additional을 추가한다. argument가 5개인 경우 fibonacci, max of four int 모두 실행하도록 구현했다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

userprog/syscall.c : 위에서 구현한 것은 system call을 인식하고 handler 함수로 보내는 과정이다. 그리고 여기서 handler 함수를 구현한다. fibonacci의 경우 47번째 수부터 int의 범위를 벗어나므로 예외처리를 진행한다. 그리고 기본적인 fibonacci 수를 구하는 방법으로 수를 반환한다. max of four int의 경우 각 2쌍씩 비교하고 큰 값들끼리 다시 비교하는 방식으로 간단하게 최댓값을 반환하도록 구현했다.

* 1. **시험 및 평가 내용**
* **fibonacci 및 max\_of\_four\_int 시스템 콜 수행 결과를 캡처하여 첨부.**
* 1. argument가 5개인 경우 (additional, num1, num2, num3, num4)



텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

9번째 피보나치 수인 34와 9, 10, 300, 400 중 가장 큰 400이 차례로 출력됐다.

* 2. argument가 2개인 경우 (additional, num)



텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

20번째 피보나치 수인 6765가 출력됐다.

* 3. argument가 음수인 경우 (additional, -num1, -num2, -num3, -num4)

****

**텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

피보나치의 범위를 넘어서면 -1을 출력하도록 했으며, 가장 큰 수인 -1이 출력됐다.