Pintos Project2

20210084 김지민, 20210216 양준영

현재 구현에 대한 분석

1. Analysis on process execution procedure
2. Analysis on system call procedure
   1. How to call syscall\_handler() in userprog/syscall.c from user program
      1. 유저 프로그램이 유저 모드에서 실행되다가 syscall을 호출하면(int $0x30을 호출하여), lib/user/syscall.c의 시스템콜을 호출한다. 이는 syscall-entry.S 파일을 호출하여 레지스터 값을 저장한다. 이후 저장한 intr\_frame을 userprog/syscall.c에 전달하며, intr\_frame의 $esp로 caller의 stack pointer를 전달한다. Caller의 스택 포인터의 32bit word에서 시스텀 호출 번호를 받아 Syscall\_handler에서 시스템 콜 번호에 해당하는 핸들러를 호출하여 처리한다. 또한, 첫 번째 인수는 스택 포인터의 다음 32bit word에 있다. 값을 반환하는 시스템 호출은 intr\_frame의 $eax를 수정하면 된다.
   2. Source code 분석(syscall.c, intr-stubs.S, interrupt.c)
      1. Syscall.c
         1. Halt: pintos를 종료하는 함수이다.
         2. Exit: 현재 프로세스를 종료시키는 syscall이다. 종료될 경우, 프로세스 이름을 출력시켜야 한다.
         3. Exec: 현재 프로세스를 cmd\_line에서 지정된 인수를 전달해 이름이 지정된 실행 파일로 바꾸는 것. 실패한다면 -1로 exit한다.
         4. Fork: 자식 프로세스를 복제 및 실행시키는 syscall이다.
         5. Wait: 자식 프로세스가 종료될 때까지 대기하는 함수로, 자식이 종료된 exit\_status를 받아온다. 만약 자식이 예외 등으로 비정상 종료가 되었다면 -1을 반환한다.
         6. Create: 파일의 이름과 경로 정보, 파일 크기를 받아 파일을 생성하는 syscall로 성공할 경우 참을 반환한다.
         7. Remove: 파일을 삭제하는 명령으로 제거할 파일의 이름과 정보를 받아 삭제한 경우 참을 반환한다.
         8. Open: 파일을 여는 함수로, 파일을 열 수 없다면 -1을, 아니면 음아 아닌 정수를 반환한다.
         9. Filesize: 열린 파일의 크기를 바이트 단위로 반환한다.
         10. Read: 열린 파일의 데이터를 읽어오는 함수이다. 열리지 않았다면 -1을 반환한다.
         11. Write: 열린 파일에 데이터를 작성하는 함수이다.
         12. Seek: 열린 파일에서 다음 파일 내 데이터의 위치를 받아 position으로 옮긴다.
         13. Tell: 열린 파일에서 읽거나 쓸 바이트의 위치를 반환한다.
         14. Close: 파일을 닫는 함수이다. 프로세스를 exit하거나 terminate하면 열린 파일을 implicit하게 닫는다.
      2. Intr-stubs.S
         1. Intr\_entry
            * Internel/External 인터럽트가 intrNN\_stub 루틴 중 하나로 실행된다. Intr\_frame 구조체, frame\_pointer, vec\_no를 스택에 push하고 여기로 점프한다. 스택에 저장하고 커널에서 필요한 레지스터를 설정하고 핸들러를 호출한다. 이후 intr\_exit을 통해 나온다.
         2. Intr\_exit
            * 호출자의 레지스터를 다시 불러오고 스택의 추가 데이터를 폐기한 후 호출자에게 반환한다.
         3. Intr\_stub
            * 256개의 stub로 각각은 해당 인터럽트 벡터의 진입점으로 사용된다.
            * 함수 포인터들의 배열인 intr\_stubs의 정확한 지점에 함수 각각의 주소를 넣는다.
            * Interrupt.c에서 intr\_handler에서 vec\_no에 따라 배열의 값에 접근해 핸들러가 실행된다.
      3. Interrupt.c
         1. Intr\_get\_level: 현재 인터럽트 상태를 반환한다.
         2. Intr\_set\_level: level로 지정한 인터럽트를 활성화(비활성화)하고 이전 인터럽트 상태를 반환한다.
         3. Intr\_enable: 인터럽트를 활성화하고 이전 인터럽트 상태를 반환한다.
         4. Intr\_disable: 인터럽트를 비활성화하고 이전 인터럽트 상태를 반환한다.
         5. Intr\_init: 인터럽트 시스템을 초기화한다.
         6. Register\_handler: privilege 단계에서 핸들러를 호출하기 위해 레지스터가 VEC\_NO를 인터럽트한다. 인터럽트 상태가 LEVEL로 설정된 상태에서 핸들러가 호출된다.
         7. Intr\_register\_ext: external interrupt VEC\_NO를 호출한다. 인터럽트를 비활성화한 상태에서 실행될 것
         8. Intr\_register\_int: internal interrupt VEC\_NO를 호출한다. 인터럽트를 비활성화한 상태에서 실행될 것
         9. Intr\_context: external 인터럽트를 처리하는 동안 참을 반환한다.
         10. Intr\_yield\_in\_return: external 인터럽트를 처리하는 동안 핸들러가 인터럽트가 반환되기 전까지 새로운 프로세스를 실행하도록 양보하는 것.
         11. Pic\_init; PIC(Programmable Interrupt Controller)를 초기화한다.
         12. Pic\_end\_of\_interrupt: 주어진 IRQ에 대해 PIC에 end-of-interrupt singal을 보낸다. 만약 IRQ를 승인하지 않으면 다시는 전달되지 않을 것
         13. Make\_gate: 함수를 호출하는 gate를 생성한다.
         14. Make\_intr\_gate: 주어진 DPL에서 함수를 호출하는 인터럽트 게이트를 호출한다.
         15. Make\_trap\_gate: 주어진 DPL에서 함수를 호출하는 trap gate를 생성한다.
         16. Make\_idtr\_operand: LIDT 명령을 위한 피연산자로 사용될 때, 주어진 LIMIT과 BASE를 양보하는 descriptor를 반환한다.
         17. Intr\_handler: 모든 인터럽트, fault, 예외를 처리하는 핸들러이다. 이 함수는 intr-stubs.S에서의 assembly 언어를 통해 호출된다.
         18. Unexpected\_interrupt: interrupt frame F와 예기치 않은 인터럽트를 처리한다. 예기치 않은 인터럽트는 등록한 핸들러가 없는 인터럽트이다.
         19. Intr\_dump\_frame: 디버깅을 위해 interrupt frame F를 console에 덤프한다.
         20. Intr\_name: 인터럽트 VEC의 이름을 반환한다.
3. Analysis on file system

목표 달성하기 위한 수정 계획

1. Process termination messages
   1. 수정할 방식
   2. 데이터 구조와 구체적 알고리즘
2. Argument passing
   1. 수정할 방식
   2. 데이터 구조와 구체적 알고리즘
3. System call
   1. 수정할 방식:
      1. System call handler: Syscall handler에서 frame의 32bit word를 읽어 핸들러 번호와 인자를 받는다. 이후 해당 핸들러에 맞는 핸들러를 호출하는 방식으로(아래의 구현한 halt와 같은 함수를 호출) syscall handler를 구현할 것이다.
      2. User process manipulation(halt, exit, exec, wait): halt의 경우는 Pintos를 종료하는 함수를 호출하고, exit은 실행 중이던 스레드를 종료하므로 thread\_exit을 통해 구현할 예정이다. Exec의 경우는 파일 이름을 받아 해당 파일을 실행 파일로 바꿀 것이다. Wait의 경우 대기를 해야하므로 semaphore를 이용하여 lock을 거는 방식으로 구현할 계획이다.
      3. File Manipulation(file system): Process처리는 process.c에 있는 함수들을 이용하여 process\_fork와 같은 함수를 사용하고, file 관리는 file system에 있는 함수들을 활용하여, filesys\_create, file\_seek, file\_tell과 같은 함수들을 사용할 수 있다면 최대한 활용하여 사용할 것이다. 파일을 동시에 작업할 수 없도록 만들어야 하므로 filesys.c, file.c에 파일을 관리하는 함수를 만들어 해당 함수를 호출하는 방식으로 구현할 예정이다.
   2. 데이터 구조와 구체적 알고리즘
      1. Syscall handler: handler 번호를 받은 후 switch 문을 이용하여 알맞은 핸들러를 호출할 예정이다.
      2. 다른 함수들의 경우 특별한 데이터 구조를 사용하진 않겠지만, lock, aquire가 필요할 수 있어 semaphore를 사용할 수 있을 것으로 생각된다.
4. Denying writes to executables
   1. 수정할 방식
   2. 데이터 구조와 구체적 알고리즘