KWANGWOON UNIVERSITY

2023년 2학기 **운영체제 & 운영체제실습**

Assignment 4

System Software Laboratory
School of Computer and Information Engineering
Kwangwoon Univ.

Requirements

- **Ubuntu 16.04.5 Desktop 64bits 환경에서 채점**
- **Copy 발견 시 0점 처리**
- **보고서**
  - **표지**
    - 수업 명, 과제 이름, 담당 교수님, 학번, 이름 필히 명시
      - 과제 이름 ➔ assignment 4
  - 아래의 내용은 보고서에 필히 포함
    - **Introduction** : 4줄 이상(background 제외) 작성
    - **Conclusion & Analysis :** 수행한 내용을 캡처 및 설명
    - **고찰** : 과제를 수행하면서 느낀점 작성
    - **Reference**
      - 과제를 수행하면서 참고한 내용을 구체적으로 기록
      - 강의자료만 이용한 경우 생략 가능

Requirements (cont'd)

- **Source**
- **아래 소스파일명과 다르게 작성하여 제출 시 감점처리됩니다**
  - **4-1**
    - file_varea.c
    - Makefile //테스트용 파일도 같이 컴파일되도록 작성
  - **4-2**
    - D_recompile.c //강의자료실에 첨부
    - D_recompile_test.c //강의자료실에 첨부
    - Makefile
- **Copy 발견 시 0점 처리**

Requirements (cont'd)

- **Softcopy만 작성(Hardcopy 받지 않음)**
- **제출 파일**
  - 보고서(.pdf) + Source file 하나의 압축파일로 압축하여 제출(**tar.gz**)
  - Tar 압축 및 해제 방법
    - 압축 시 ➔ tar –zcvf [압축 파일명].tar.gz[폴더 명]
    - 해제 시 ➔ tar –zxvf 파일명.tar.gz
- **보고서 및 압축 파일 명 양식**
  - **os_과제번호_학번_수강분류코드**

수강요일	이론_화목	이론_금12	실습
수강분류코드	A	B	C

  - e.g
    - 이론_월수 수강하는 학생인 경우
      (보고서) os_4_20221234567_A.pdf (압축파일) os_4_20221234567_A.tar.gz
    - 이론_금12 수강하는 학생인 경우
      (보고서) os_4_20221234567_B.pdf (압축파일) os_4_20221234567_B.tar.gz
    - 실습 수강하는 학생인 경우
      (보고서) os_4_20221234567_C.pdf (압축파일) os_4_20221234567_C.tar.gz

Requirements (cont'd)

- **실습 수업을 수강하는 학생인 경우**
  - 실습 과목에 과제를 제출(.tar.gz)
  - 이론 과목에 간단한 .txt 파일로 제출

    - **이론 과목에 .txt 파일 미 제출 시 감점**

- **과제 제출**
  - **KLAS – 강의 과제 제출**
  - **2023년 11월 23일 목요일 23:59까지 제출**
    - **딜레이 받지 않음**
      - 제출 마감 시간 내 미제출시 해당 과제 0점 처리

Assignment 4-1: **Files located in VM Area**

- **PID를 바탕으로 아래와 같은 프로세스 정보를 출력하는 Module 작성**
  - **프로세스의 이름과 pid**
  - 정보가 위치하는 **가상 메모리 주소**
  - 프로세스의 **데이터 주소, 코드 주소, 힙 주소**
  - 정보의 원본 파일의 **전체 경로**

  - 프로세스의 가상 메모리에는 파일에서 불러온 여러 정보가 적재되어 있다.
    - e.g. shared library, binary file…

- **Requirements**
  - 2차 과제에서 작성한 ftrace 시스템 콜(336번)을 다음 함수로 wrapping 하여 사용
  - **Hooking 함수 명: file_varea**

- **Hints**
  - 관련 강의자료를 참고하여 struct task_struct 부터 분석
  - 관련 강의자료의 예시 코드를 참고
  - pid_task()

Assignment 4-1

- **Examples**

```
sslab@ubuntu:~/assign4/4_1$ sudo insmod file_varea.ko
sslab@ubuntu:~/assign4/4_1$ ./assin4
sslab@ubuntu:~/assign4/4_1$ sudo rmmod file_varea
sslab@ubuntu:~/assign4/4_1$ dmesg
```

```
######## Loaded files of a process 'assin4(45339)' in VM ########
mem[400000~401000] code[400000~40074c] data[600e10~601040] heap[1fe9000~1fe9000] /home/sslab/assign4/4_1/assin4
mem[600000~601000] code[400000~40074c] data[600e10~601040] heap[1fe9000~1fe9000] /home/sslab/assign4/4_1/assin4
mem[601000~602000] code[400000~40074c] data[600e10~601040] heap[1fe9000~1fe9000] /home/sslab/assign4/4_1/assin4
mem[7f4d3f41a000~7f4d3f5da000] code[400000~40074c] data[600e10~601040] heap[1fe9000~1fe9000] /lib/x86_64-linux-gnu/libc-2.23.so
mem[7f4d3f5da000~7f4d3f7da000] code[400000~40074c] data[600e10~601040] heap[1fe9000~1fe9000] /lib/x86_64-linux-gnu/libc-2.23.so
mem[7f4d3f7da000~7f4d3f7de000] code[400000~40074c] data[600e10~601040] heap[1fe9000~1fe9000] /lib/x86_64-linux-gnu/libc-2.23.so
mem[7f4d3f7de000~7f4d3f7e0000] code[400000~40074c] data[600e10~601040] heap[1fe9000~1fe9000] /lib/x86_64-linux-gnu/libc-2.23.so
mem[7f4d3f7e4000~7f4d3f80a000] code[400000~40074c] data[600e10~601040] heap[1fe9000~1fe9000] /lib/x86_64-linux-gnu/ld-2.23.so
mem[7f4d3fa09000~7f4d3fa0a000] code[400000~40074c] data[600e10~601040] heap[1fe9000~1fe9000] /lib/x86_64-linux-gnu/ld-2.23.so
mem[7f4d3fa0a000~7f4d3fa0b000] code[400000~40074c] data[600e10~601040] heap[1fe9000~1fe9000] /lib/x86_64-linux-gnu/ld-2.23.so
################################################################
```

Assignment 4-2: **Dynamic Recompilation**

Dynamic Recompilation

- **Dynamic recompilation** is a feature of some **emulators and virtual machines**, where the system may recompile some part of a program during execution.
- By compiling during execution, the system can tailor the generated code to reflect the program's run-time environment, and potentially produce more efficient code by exploiting information.

- Usages
  - JVM (JIT), objdump
  - QEMU, VirtualBox, …

Assignment 4-2 **Dynamic Recompilation**

- **Problems**
  - A, B 두 기기에서 모두 실행할 수 있는 실행파일 있음
  - A는 먼저 나온 모델로, add instruction이 1씩 더하기 가능
  - B는 나중에 나온 모델로, add instruction이 확장됨.
    - 1이 아닌 숫자도 더하기 가능

  - 실행파일의 코드의 add instruction은 모두 1씩 더하는 코드

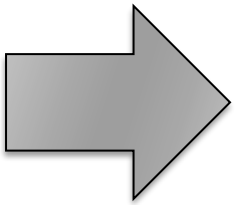
```
int add(int a)
{
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        return a;
}
```

KWANGWOON UNIVERSITY

Assignment 4-2: **Dynamic Recompilation**

- **Problems**
  - B에서도 해당 파일을 실행 가능
    - 코드가 있다면 add instruction 이 포함된 부분을 다음과 같이 **최적화 가능**
      - 1이 아닌 다른 숫자를 이용한 add instruction을 수행할 수 있음

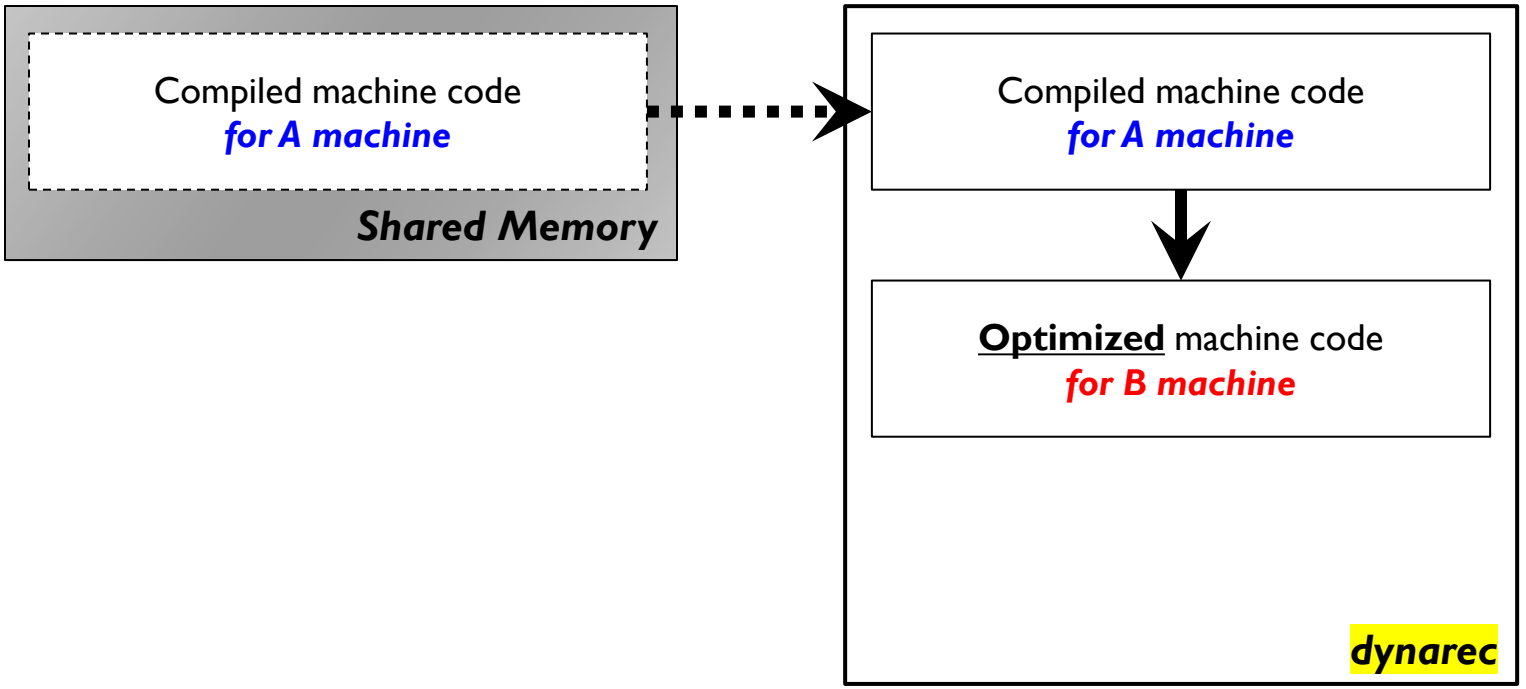
```
int add(int a)
{
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        a++;
        return a;
}
```



```
int add(int a)
{
        return a+20;
}
```

Assignment 4-2: **Dynamic Recompilation**

- **Problems**
  - 하지만, 우리에겐 코드 없음
    - 따라서, 컴파일 된 코드를 최적화 하는 D_recompile을 작성 필요



Assignment 4-2: **Dynamic Recompilation**

- **주의 사항**
  - 매 실험 전에 아래의 명령어를 수행할 것
    - 캐시 및 버퍼를 비워서 실험에 영향을 주는 요소를 제거

$ **sync**
- Linux command to flush file system buffer

$ **echo 3 | sudo tee /proc/sys/vm/drop_caches**
- Linux commands to free pagecache, dentries, and inodes

Assignment 4-2: **Dynamic Recompilation**

- **To-do List**
  - **1. Shared memory에서 컴파일 된 코드에 접근**
  - **2. Code section의 함수는 마음대로 수정 할 수 없음**
    - 해당 영역에 write권한이 없기 때문 (i.e. r-x)
    - Write 권한이 있는 영역을 할당 (uint8_t *compiled_code, 전역 변수)
    - 그 영역에 함수를 복사 (execute 권한 없음, i.e. rw-)
  - **3. 복사한 함수를 최적화**
    - **하드코딩 방지**
    - **최적화할 명령어는 add, sub, imul, div**
    - 0 division, -로 인한 음수에 대한 예외는 없다고 가정
    - 연산이 중복으로 나올 경우 하나로 합치기
    - [ex]
      - add instruction이 중복으로 나오면 합치기 (최적화)
      - sub instruction이 중복으로 나오면 합치기 (최적화)
      - multiple instruction이 중복으로 나오면 합치기 (최적화)
      - Division instructio이 중복으로 나오면 합치기 (최적화)

Assignment 4-2: **Dynamic Recompilation**

- **4. 수정한 함수를 실행하자.**
  - 수정한 함수에는 execute 권한이 없으므로
  - 권한 변경: r-x
- **5. 컴파일한 코드와 기존 코드의 실행 시간을 비교**
  - 보고서에 명시
  - **50 set 실험한 값을 표로 나타내고, 그 값에 평균을 취할 것**
    - **1 set**
      1. 캐시 및 버퍼 지우기
      2. 최적화 하기 전 결과
      3. 캐시 및 버퍼 지우기
      4. 최적화 후 결과

Assignment 4-2: **Dynamic Recompilation**

- **보고서에 추가 작성**
  - objdump 뜨는 과정
  - dump 뜬 파일의 화면
  - 문제 해결과정
    - dump 뜬 파일로 부터 어떻게 문제 해결의 실마리를 잡았는지

- **objdump 사용법**
  - gcc –c D_recompile_test.c
    ➔ object file (D_recompile_test.o) 생성
  - $ objdump –d [object file]
    ex) $ objdump –d D_recompile_test.o

```
sslab@ubuntu:~/assign4/4_2$ ls
D_recompile.c  D_recompile_test.c  Makefile
sslab@ubuntu:~/assign4/4_2$ gcc -c D_recompile_test.c
sslab@ubuntu:~/assign4/4_2$ objdump -d D_recompile_test.o > test
sslab@ubuntu:~/assign4/4_2$ ls
D_recompile.c  D_recompile_test.c  D_recompile_test.o  Makefile  test
sslab@ubuntu:~/assign4/4_2$
```

- **Redirection**
  - objdump 결과를 file로 저장

ex) $ objdump –d D_recompile_test.o > test

Assignment 4-2: **Dynamic Recompilation**

- **Restriction**
  - 메모리 할당: mmap() 사용
  - 할당한 메모리의 권한은 write와 execute를 동시에 부여하지 않음

  - **미리 구현된 add(),suv(),imul(),div() 함수는 recompile: D_recompile_test.c 참고**
    ➔ 수정해서 사용 가능
  - **제공하는 파일을 수정하여 완성: D_recompile.c 참고**

  - DIV 연산 시 register는 dl 을 사용
    - 값은 고정

  - Makefile 규칙
    - $ make : recompile 하지 않는 기존 코드로 컴파일
    - $ make dynamic : recompile 하는 코드로 컴파일
    - Hint: #ifdef, #endif

Makefile 형식 예시

```
EXEC = D_recompile
cc = gcc

default:
	$(CC) -o drecompile D_recompile.c

dynamic:
	$(CC) -Ddynamic -o drecompile D_recompile.c

clean:
	rm -rf D_recompile $(EXEC)
```

Assignment 4-2: **Dynamic Recompilation**

- **결과 화면**
  - 예시

```
sslab@ubuntu:~/assign4/4_2$ sync
sslab@ubuntu:~/assign4/4_2$ echo 3| sudo tee /proc/sys/vm/drop_caches
3
sslab@ubuntu:~/assign4/4_2$ gcc -o test2 D_recompile_test.c
sslab@ubuntu:~/assign4/4_2$ ./test2
Data was filled to shared memory.
sslab@ubuntu:~/assign4/4_2$ make
cc -o drecompile D_recompile.c
sslab@ubuntu:~/assign4/4_2$ ./drecompile
total execution time: 0.29085 sec
sslab@ubuntu:~/assign4/4_2$
sslab@ubuntu:~/assign4/4_2$ sync
sslab@ubuntu:~/assign4/4_2$ echo 3| sudo tee /proc/sys/vm/drop_caches
3
sslab@ubuntu:~/assign4/4_2$ ./test2
Data was filled to shared memory.
sslab@ubuntu:~/assign4/4_2$ make dynamic
cc -Ddynamic -o drecompile D_recompile.c
sslab@ubuntu:~/assign4/4_2$ ./drecompile
total execution time: 0.23421591 sec
sslab@ubuntu:~/assign4/4_2$
```

최적화하기 전 결과

최적화한 결과