Systems Programming

Spring 2023

Q&A Session 진행 계획

- **1주차 (3/9 목)**
  - 01-linking
  - 02-relocation
- **2주차 (3/16 목)**
  - 03-exceptions
  - 04-signals
- **3주차 (3/23 목)**
  - 04-signals
  - 05-io
- **4주차 (3/30 목)**
  - 06-vm-concepts
  - 07-vm-systems
- **Tentative (4/4 화)**
- **중간고사1 (4/6 목)**

내용

- 각 주차 강의내용 review & summary
- Q&A collected by google forms
- Supplements will be uploaded through eTL

1 주차

- Review & Summary
  - 01-linking
  - 02-relocation

- Q&A

Questions

1.
01-linking.pdf 38p의 3번 예시에서 int x와 double x를 모두 weak symbol로 정의하면 둘 중 하나를 OS가 임의로 선택하기 때문에 절반의 확률로 double이 선택될 경우 y를 덮어쓸 가능성이 있다고 이해했는데 맞게 이해했을까요?
아니면 int로 선언한 이후 double로 재선언되었기 때문에 현재 x는 double형태로 인식되며 int범위 내의 숫자가 들어오면 y가 영향을 받지 않지만 이보다 큰 숫자가 들어오면 y가 덮어씌워질 수 있다는 의미인가요?

2.
01-linking.pdf 38p의 4번째 예시를 보면 p1()이 있는 파일에서 strong symbol로 int x=7을 정의했으며 p2()가 있는 파일에서는 weak symbol로 double x를 정의했는데, 오른쪽 해설을 보면 x에 어떤 숫자를 넣게 된 경우 y를 overwrite 한다고 되어있습니다.
하지만 strong과 weak가 동시에 있는 경우 strong을 우선적으로 사용하기 때문에 int로 정의된 x의 값이 바뀌게 되며, 만약 int 범위를 넘어가는 값을 x에 넣으려 했다면 오버플로우가 일어나서 다른 값이 저장될 뿐, y에는 영향이 없어야 하는것 아닌가요?
결국 1,2번 질문 모두 int랑 double로 같은 변수를 선언했을 때 컴퓨터가 어떤 방식으로 받아들이는지 잘 이해가 가지 않아 질문드립니다.

Linker Puzzles

1.

01-linking.pdf 38p의 3번 예시에서 int x와 double x를 모두 weak symbol로 정의하면 둘 중 하나를 OS가 임의로 선택하기 때문에 절반의 확률로 double이 선택될 경우 y를 덮어쓸 가능성이 있다고 이해했는데 맞게 이해했을까요? 아니면 int로 선언한 이후 double로 재선언되었기 때문에 현재 x는 double형태로 인식되며 int범위 내의 숫자가 들어오면 y가 영향을 받지 않지만 이보다 큰 숫자가 들어오면 y가 덮어씌워질 수 있다는 의미인가요?

```
int x;
p1() {}
```

```
int x;
p2() {}
```

References to **x** will refer to the same uninitialized int. Is this what you really want?

```
int x;
int y;
p1() {}
```

```
double x;
p2() {}
```

Writes to **x** in **p2** might overwrite **y**! Evil!

```
int x=7;
int y=5;
p1() {}
```

```
double x;
p2() {}
```

Writes to **x** in **p2** will overwrite **y**! Nasty!

```
int x=7;
p1() {}
```

```
int x;
p2() {}
```

References to **x** will refer to the same initialized variable.

Nightmare scenario: two identical weak structs, compiled by different compilers with different alignment rules.

2.
01-linking.pdf 38p의 4번째 예시를 보면 p1()이 있는 파일에서 strong symbol로 int x=7을 정의했으며 p2()가 있는 파일에서는 weak symbol로 double x를 정의했는데, 오른쪽 해설을 보면 x에 어떤 숫자를 넣게 된 경우 y를 overwrite 한다고 되어있습니다.
하지만 strong과 weak가 동시에 있는 경우 strong을 우선적으로 사용하기 때문에 int로 정의된 x의 값이 바뀌게 되며, 만약 int 범위를 넘어가는 값을 x에 넣으려 했다면 오버플로우가 일어나서 다른 값이 저장될 뿐, y에는 영향이 없어야 하는것 아닌가요?
결국 1,2번 질문 모두 int랑 double로 같은 변수를 선언했을 때 컴퓨터가 어떤 방식으로 받아들이는지 잘 이해가 가지 않아 질문드립니다.

```
int x;
int y;
p1() {}
```

```
double x;
p2() {}
```

Writes to **x** in **p2** might overwrite **y**!
Evil!

```
int x=7;
int y=5;
p1() {}
```

```
double x;
p2() {}
```

Writes to **x** in **p2** will overwrite **y**!
Nasty!

```
int x=7;
p1() {}
```

```
int x;
p2() {}
```

References to **x** will refer to the same initialized variable.

Nightmare scenario: two identical weak structs, compiled by different compilers with different alignment rules.

Questions

3.
01-linking.pdf 51p에서 run-time linking에 관한 설명에 보면 Distributing software라는 문장이 있는데 교수님 강의에서 정확히 설명을 해주시지 않아 잘 이해하지 못해 질문드립니다!

Dynamic linking is a powerful and useful technique. Here are some examples in the real world:

- *Distributing software.* Developers of Microsoft Windows applications frequently use shared libraries to distribute software updates. They generate a new copy of a shared library, which users can then download and use as a replacement for the current version. The next time they run their application, it will automatically link and load the new shared library.

Questions

linking - 4 강의에 대해서 질문있습니다.
강의자료 01-linking 의 p.54 내용을 설명하시며 교수님께서 dlclose(handle) 이 실행되고 나서도 addvec은 메모리에 존재한다고 하셨는데, 이는 libvector.so의 reference count가 0이 되지 않은 경우를 가정하고 말씀하신 것인지 궁금합니다.
만약 dlclose(handle)을 통해 libvector.so의 reference count가 1에서 0이 된다면 해당 shared library는 unload 되지 않을까 하는 의문이 들어서 질문드리게 되었습니다.

dlclose()

The function dlclose() **decrements the reference count on the dynamically loaded shared object referred to by handle**. **If the reference count drops to zero, then the object is unloaded.** All shared objects that were automatically loaded when dlopen() was invoked on the object referred to by handle are recursively closed in the same manner.

```
#include <dlfcn.h>

int dlclose (void *handle);
```

Returns: 0 if OK, −1 on error

The `dlclose` function unloads the shared library if no other shared libraries are still using it.

Questions

02-relocation 강의 녹음 파일 17:20에서 교수님께서 static이기 때문에 .bss로 가게 된다 하셨는데, 초기화가 되어있지 않기 때문에 (정확히는 static의 초기화를 하지 않아 0으로 초기화가 되어있기 때문에) .bss로 가게 된다는 게 맞지 않을까요?
static int *bufp1 = &buf[1]; 는 불가능하겠지만 만약 static int t = 3; 이라면 t는 분명 .data에 있어야 할 것 같습니다.

- .bss section
  - NOT initialized, initialized to zero (implicit initialization)
- .data section
  - Initialized to a non-zero value

```c
#include <stdio.h>

static int i;

int f() {
    i++;
    return i;
}

int main() {
    printf("%d\n", f());
    printf("%d\n", f());

    return 0;
}
```

```
Disassembly of section .text:

0000000000000000 <f>:
#include <stdio.h>

static int i;

int f() {
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
    i++;
   4:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # a <f+0xa>
                        6: R_X86_64_PC32        .bss-0x4
   a:   83 c0 01                add    $0x1,%eax
   d:   89 05 00 00 00 00       mov    %eax,0x0(%rip)        # 13 <f+0x13>
                        f: R_X86_64_PC32        .bss-0x4
    return i;
  13:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # 19 <f+0x19>
                        15: R_X86_64_PC32       .bss-0x4
}
  19:   5d                      pop    %rbp
  1a:   c3                      retq

000000000000001b <main>:
```

```c
#include <stdio.h>

static int i = 3;

int f() {
    i++;
    return i;
}

int main() {
    printf("%d\n", f());
    printf("%d\n", f());

    return 0;
}
```

```
Disassembly of section .text:

0000000000000000 <f>:
#include <stdio.h>

static int i = 3;

int f() {
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
    i++;
   4:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # a <f+0xa>
                        6: R_X86_64_PC32        .data-0x4
   a:   83 c0 01                add    $0x1,%eax
   d:   89 05 00 00 00 00       mov    %eax,0x0(%rip)        # 13 <f+0x13>
                        f: R_X86_64_PC32        .data-0x4
    return i;
  13:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # 19 <f+0x19>
                        15: R_X86_64_PC32       .data-0x4
}
  19:   5d                      pop    %rbp
  1a:   c3                      retq

000000000000001b <main>:
```

Questions

Exception 8쪽에서 interrupts는 handler returns to "next" instruction라고 나와있는데, 프로세스를 종료하는 ctrl-c나 soft/hard reset에서도 유효한 이야기인가요?

Questions

그리고 혹시 주차에 관계없이 질문드려도 상관없을까요?
(다음주에 5주차 내용인 io관련 내용을 질문한다던가, 아니면 시험전주에 linking에 관한 질문을 한다던가)