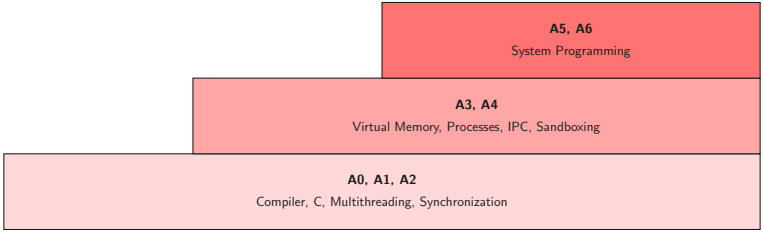
System Level Programming

Daniel Gruss

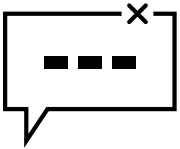
2021-03-15

Course Overview



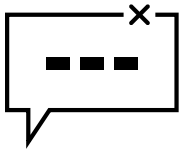
A3 - Virtual Memory

Segmentation fault



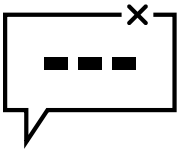
- We've all been there: access to "invalid" memory location

Segmentation fault



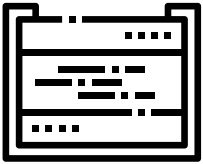
- We've all been there: access to "invalid" memory location
- But aren't pointers indices of this large array called RAM / physical memory?

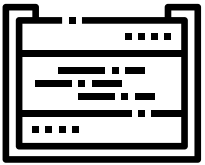
Segmentation fault



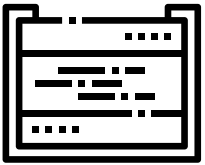
- We've all been there: access to "invalid" memory location
- But aren't pointers indices of this large array called RAM / physical memory?
- How can addresses in physical memory be "invalid"?

- **Pointers are not addresses/indices in a large array called RAM / physical memory**

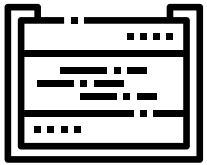




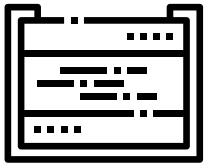
- **Pointers are not addresses/indices in a large array called RAM / physical memory**
- **but** in a large array called **virtual memory**



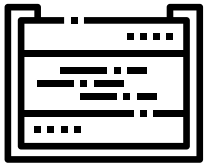
- **Pointers are not addresses/indices in a large array called RAM / physical memory**
- **but** in a large array called **virtual memory**
- There is a **big map** to translate pointers (virtual addresses) to actual physical addresses



- **Pointers are not addresses/indices in a large array called RAM / physical memory**
- **but** in a large array called **virtual memory**
- There is a **big map** to translate pointers (virtual addresses) to actual physical addresses
- In SLP / as a userspace programmer: we **never** see actual physical addresses - only pointers / virtual addresses!



- **Pointers are not addresses/indices in a large array called RAM / physical memory**
- **but** in a large array called **virtual memory**
- There is a **big map** to translate pointers (virtual addresses) to actual physical addresses
- In SLP / as a userspace programmer: we **never** see actual physical addresses - only pointers / virtual addresses!
- mapping block-wise is easier: mapping a block aka **page**



- **Pointers are not addresses/indices in a large array called RAM / physical memory**
- **but** in a large array called **virtual memory**
- There is a **big map** to translate pointers (virtual addresses) to actual physical addresses
- In SLP / as a userspace programmer: we **never** see actual physical addresses - only pointers / virtual addresses!
- mapping block-wise is easier: mapping a block aka **page**

→ different processes can use the same pointer / virtual address, but "see" different things there

- Experiment with different kinds of variables, which addresses do they get?

- Experiment with different kinds of variables, which addresses do they get?
- Observe memory usage in practice, when does it really increase?

- Experiment with different kinds of variables, which addresses do they get?
- Observe memory usage in practice, when does it really increase?
- **Answer questions from the test system questionnaire!**

- Experiment with different kinds of variables, which addresses do they get?
- Observe memory usage in practice, when does it really increase?
- **Answer questions from the test system questionnaire!**
- **Register + participate in one of the virtual memory discussions!**

- Experiment with different kinds of variables, which addresses do they get?
- Observe memory usage in practice, when does it really increase?
- **Answer questions from the test system questionnaire!**
- **Register + participate in one of the virtual memory discussions!**
- Change due to Virtual Semester: Less of a discussion, more like a "Kreuzerlübung" $\rightarrow$ undo any answers

A3 - Changes due to Virtual Setting

- Less of a discussion - more like a “Kreuzerlübung”

A3 - Changes due to Virtual Setting

- Less of a discussion - more like a "Kreuzerlübung"
- How to proceed? (if you ignore this you won't be able to answer all of A3)

A3 - Changes due to Virtual Setting

- Less of a discussion - more like a "Kreuzerlübung"
- How to proceed? (if you ignore this you won't be able to answer all of A3)
  1. Answer **all** questions, question by question

A3 - Changes due to Virtual Setting

- Less of a discussion - more like a "Kreuzerlübung"
- How to proceed? (if you ignore this you won't be able to answer all of A3)
  1. Answer **all** questions, question by question
  2. Before the deadline: **undo** all answers you don't feel confident presenting, explaining your solution in front of the class

A3 - Changes due to Virtual Setting

- Less of a discussion - more like a "Kreuzerlübung"
- How to proceed? (if you ignore this you won't be able to answer all of A3)
  1. Answer **all** questions, question by question
  2. Before the deadline: **undo** all answers you don't feel confident presenting, explaining your solution in front of the class
- "I guessed correctly" $\rightarrow$ not sufficient

A3 - Changes due to Virtual Setting

- Less of a discussion - more like a "Kreuzerlübung"
- How to proceed? (if you ignore this you won't be able to answer all of A3)
  1. Answer **all** questions, question by question
  2. Before the deadline: **undo** all answers you don't feel confident presenting, explaining your solution in front of the class
- "I guessed correctly" $\rightarrow$ not sufficient
- "I read online that this is the answer" $\rightarrow$ not sufficient

A3 - Changes due to Virtual Setting

- Less of a discussion - more like a "Kreuzerlübung"
- How to proceed? (if you ignore this you won't be able to answer all of A3)
  1. Answer **all** questions, question by question
  2. Before the deadline: **undo** all answers you don't feel confident presenting, explaining your solution in front of the class
- "I guessed correctly" $\rightarrow$ not sufficient
- "I read online that this is the answer" $\rightarrow$ not sufficient
- We want a full explanation for the answer and **what I have to do to observe the behavior you describe**

A3 - Changes due to Virtual Setting

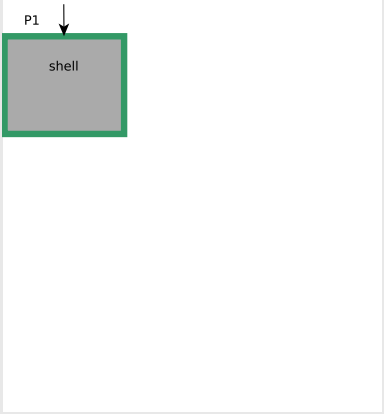
- Less of a discussion - more like a "Kreuzerlübung"
- How to proceed? (if you ignore this you won't be able to answer all of A3)
  1. Answer **all** questions, question by question
  2. Before the deadline: **undo** all answers you don't feel confident presenting, explaining your solution in front of the class
- "I guessed correctly" $\rightarrow$ not sufficient
- "I read online that this is the answer" $\rightarrow$ not sufficient
- We want a full explanation for the answer and **what I have to do to observe the behavior you describe**
- Don't collaborate with others - we cross check who did what when, answered which question when, etc.

A4 - Interprocess Communication

Code

```
//shell stuff
```

Image

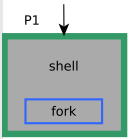


Code

```
//shell stuff

pid_t pid = fork();
```

Image

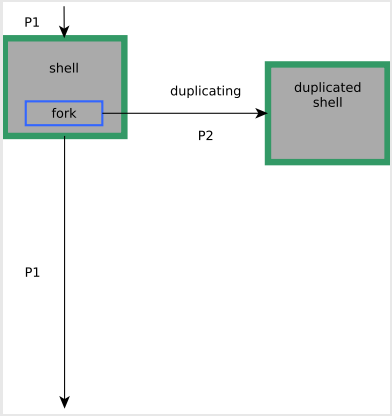


Code

```
//shell stuff

    pid_t pid = fork();
    if(pid == 0)
```

Image

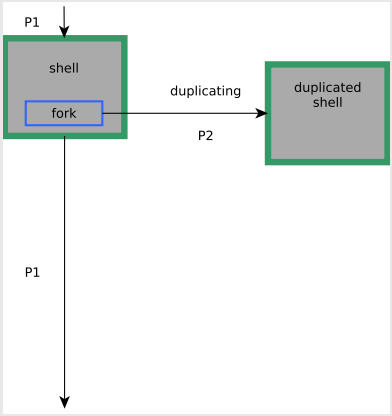


Code

```
//shell stuff

    pid_t pid = fork();
    if(pid == 0)
    {
            const char* args[] = {"~/
                "};
    }
    else
    {
            //do further shell stuff
    }
```

Image

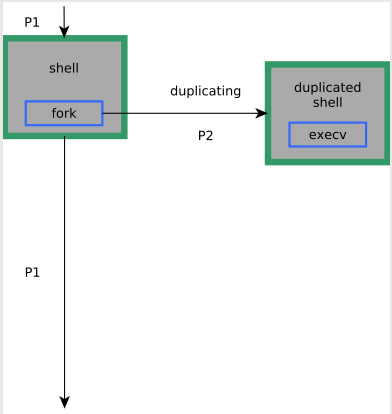


Code

```
//shell stuff

pid_t pid = fork();
if(pid == 0)
{
  const char* args[] = {"~/"};
  execv("/bin/ls", args);
}
else
{
  //do further shell stuff
}
```

Image

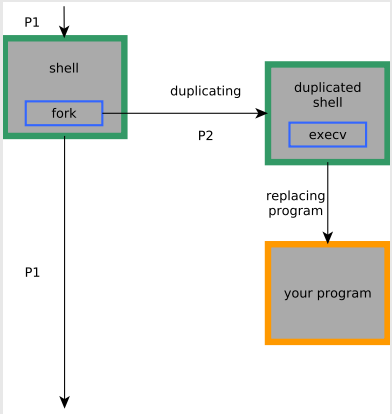


Code

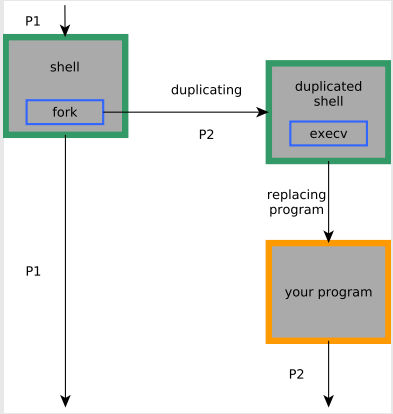
```
//shell stuff

pid_t pid = fork();
if(pid == 0)
{
  const char* args[] = {"~/"};
  execv("/bin/ls", args);
}
else
{
  //do further shell stuff
}
```

Image



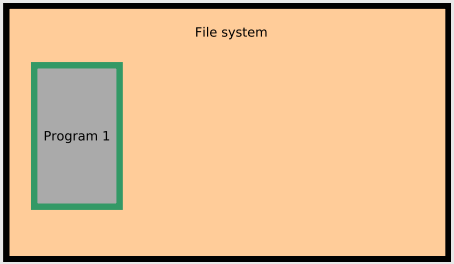
Image


P1
shell
fork
duplicating
P2
duplicated
shell
execv
replacing
program
your program
P1
P2

Code

```
/* just the start of the main */
```

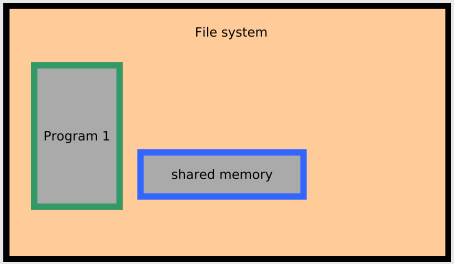
Image



Code

```
/* found in (/dev/shm/obj)  */
int fd = shm_open("obj",O_RDWR,0644);
```

Image

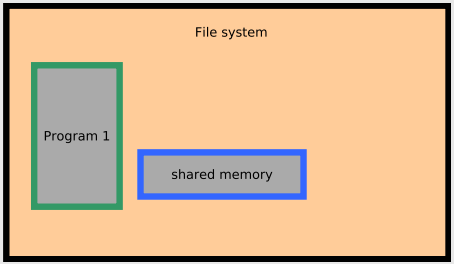


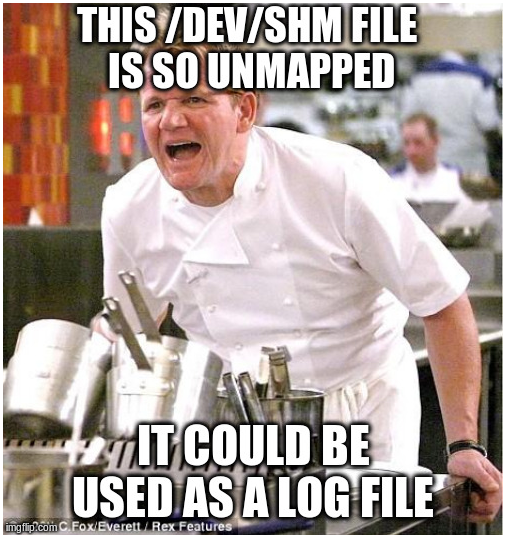
Code

```
/* found in (/dev/shm/obj)  */
int fd = shm_open("obj",O_RDWR,0644);

/* enlarge the shared memory object */
ftruncate(fd, 1000);
```

Image




THIS /DEV/SHM FILE
IS SO UNMAPPED
IT COULD BE
USED AS A LOG FILE
imgflip.com © Fox/Everett / Rex Features


WHY DON'T WE
TAKE THE SHM FILE
AND MAP IT INTO
THE VIRTUAL MEMORY

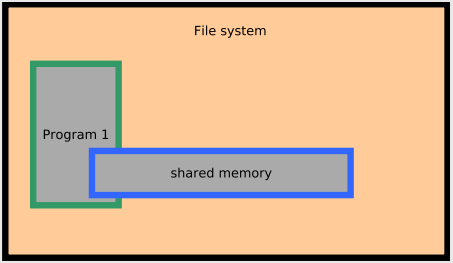
Code

```
/* found in (/dev/shm/obj)  */
int fd = shm_open("obj",O_RDWR,0644);

/* enlarge the shared memory object */
ftruncate(fd, 1000);

/* now map the shared object */
char* ptr = (char*) mmap(NULL, 1000,
    PROT_READ | PROT_WRITE, MAP_SHARED, fd,
    0);
```

Image



Code

```
/* found in (/dev/shm/obj)  */
int fd = shm_open("obj",O_RDWR,0644);

/* enlarge the shared memory object */
ftruncate(fd, 1000);

/* now map the shared object */
char* ptr = (char*) mmap(NULL, 1000,
    PROT_READ | PROT_WRITE, MAP_SHARED, fd,
    0);

/* fork the process */
pid_t pid = fork();
```

Image