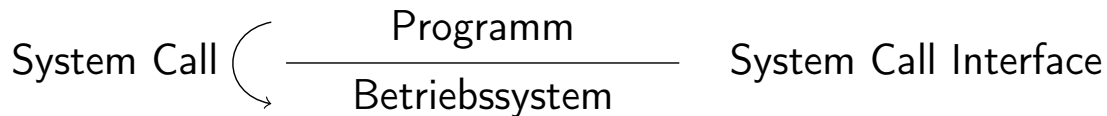


System Calls

- ▶ Aus Sicherheitsgründen: Programme unterliegen Restriktionen
 - ▶ Beispiel: Lesen/Schreiben von Dateien
- ▶ Zugriff geregelt über Anfragen an das Betriebssystem – *System Calls*
 - ▶ Interface zwischen Programm und Betriebssystem
 - ▶ System Call $\hat{=}$ Funktionsaufruf
 - ▶ Werden eindeutig über System Call Numbers identifiziert



System Calls auf x86-64 Linux Systemen

- ▶ Instruktion `syscall`
- ▶ Übergibt Kontrolle an den Kernel des Betriebssystems
- ▶ Danach wird normale Programmausführung fortgesetzt

Wichtige Referenzen:

- ▶ System V ABI
- ▶ Intel Software Developer's Manual: `syscall`

Quiz: System Calls (1)

In welchen Registern und in welcher Reihenfolge werden Argumente an System Calls übergeben?

☐

`rdi, rsi, rdx, rcx, r8 und r9`

☐

`rdi, rsi, rdx, r8, r9 und r10`

☐

`rdi, rsi, rcx, r10, r8 und r9`

☐

`rdi, rsi, rdx, r10, r8 und r9`

Quiz: System Calls (2)

Wo wird die Nummer des auszuführenden System Calls übergeben?

☐

Im Register rax

☐

Im Register rcx

☐

Im Register r11

☐

Auf dem Stack

Quiz: System Calls (3)

In welchem Register steht der Rückgabewert eines System Calls?

☐

rdi

☐

rax

☐

rcx

☐

r11

Quiz: System Calls (4)

Welche der folgenden System Call Rückgabewerte zeigen an, dass beim Ausführen eines System Calls mit der `syscall` Instruktion ein Fehler aufgetreten ist?

☐

0

☐

-4096

☐

-1

☐

0x7FFF FFFF FFFF FFFF ($2^{64} - 1$)

☐

-4095

☐

0x8000 0000 0000 0000 (-2^{64})

Quiz: System Calls (5)

Welche Register können durch einen System Call verändert werden?

Hinweis: Bedenken Sie, dass auch die Instruktion `syscall` selbst Register verändern kann.

☐

`rax`

☐

`rdx`

☐

`rbx`

☐

`r10`

☐

`rcx`

☐

`r11`

Quiz: System Calls (6)

Welcher Wert wird von der Instruktion `syscall` in das Register `rcx` geschrieben?

☐

Die UID des Nutzers, der das Programm ausführt

☐

Die Adresse, an der die Programmausführung i.A. nach dem System Call fortgesetzt wird

☐

Das Privilege Level des Programms

☐

Der Wert von `rflags` vor dem System Call

Layout einer Programmbinary

- ▶ Programme liegen als ELF (**E**xecutable and **L**inkable **F**ormat) Datei vor
- ▶ Enthält: Programmcode, und Programm- und Metadaten

Wichtige Informationen:

- ▶ Adressen an welche Segmente geladen werden sollen → Program-Header
- ▶ Startadresse der Programmausführung (`_start`) → Datei-Header

Layout einer Programmbinary

```
$ readelf -hW testprog
```

```
ELF Header:
```

```
...
```

```
Machine:                        Advanced Micro Devices X86-64
```

```
...
```

```
Entry point address:           0x401000
```

```
...
```

```
Program Headers:
```

Type	Offset	VirtAddr	PhysAddr	FileSiz	MemSiz	Flg	Align
LOAD	0x000000	0x0000000000400000	0x0000000000400000	0x00017c	0x00017c	R	0x1000
LOAD	0x001000	0x0000000000401000	0x0000000000401000	0x00000b	0x00000b	R E	0x1000
LOAD	0x002000	0x0000000000402000	0x0000000000402000	0x000038	0x000038	R	0x1000
NOTE	0x000158	0x0000000000400158	0x0000000000400158	0x000024	0x000024	R	0x4
GNU_STACK	0x000000	0x0000000000000000	0x0000000000000000	0x000000	0x000000	RW	0x10

Start eines Programms

- ▶ Mittels System Call: `execve`
- ▶ Bei erfolgreicher Ausführung: Kein Rückgabewert
- ▶ Aktuelles Programm wird durch neues “ersetzt”
- ▶ Binary wird in Speicher geladen und Stack initialisiert

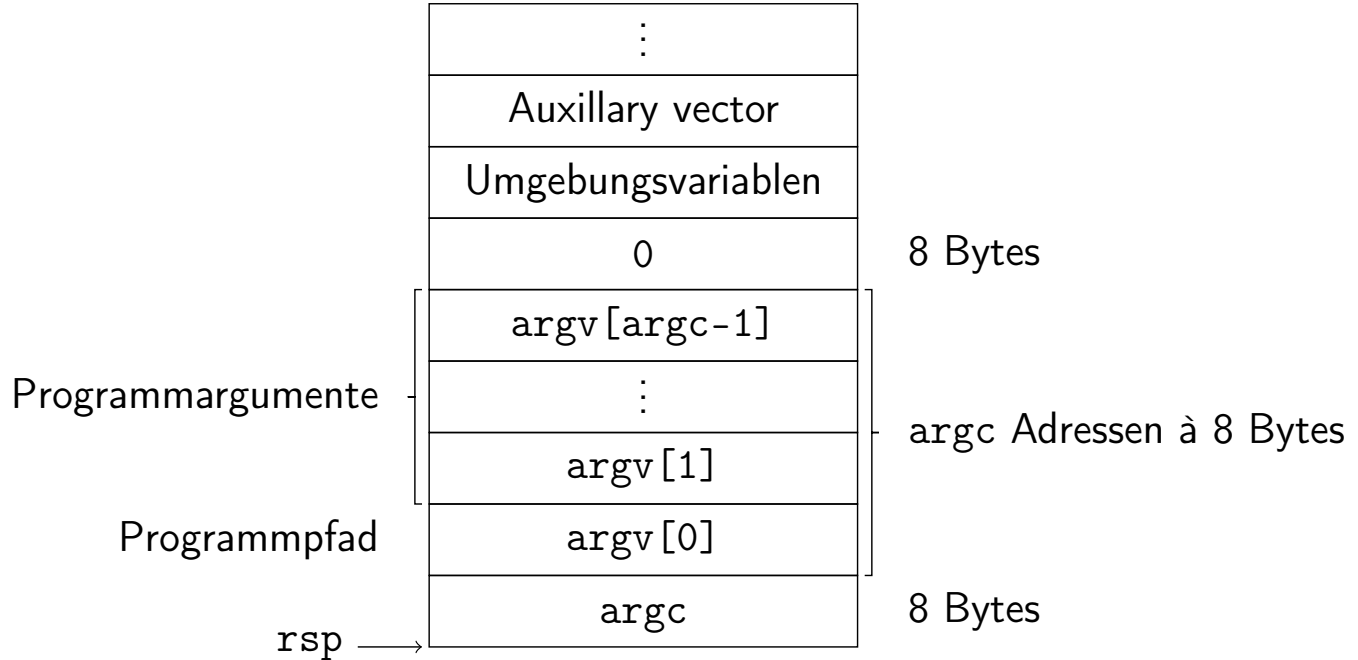
Start eines Programms

Daten zu Programmstart (auf dem Stack):

- ▶ Programmargumente
- ▶ Umgebungsvariablen
- ▶ Auxilliary Vector

```
$ ./testprog arg1 arg2
```

Start eines Programms



Textausgabe auf der Konsole

Mit dem System Call `write`, und `stdout`.

Hilfreiche Referenzen:

- ▶ `man 2 write`
- ▶ `man 3 stdout`
- ▶ Eine System Call Number Referenz

Quiz: write (1)

Was ist die System Call Number des write System Calls auf x86-64?

☐

0

☐

1

☐

2

☐

0xFFFF 0001

Quiz: write (2)

Wie viele Parameter erwartet der `write` System Call?

☐

1

☐

2

☐

3

☐

4

Quiz: write (3)

Welcher File Descriptor entspricht standardmäßig dem Stream stdout?

☐

0

☐

1

☐

2

☐

0xFFFF 0001

Quiz: write (4)

Welchen Rückgabewert hat der System Call `write(1,buf,15)`, sofern dieser erfolgreich ausgeführt wurde?

☐

0

☐

1

☐

Einen Wert zwischen 1 und 15

☐

15

Beenden eines Programms

- ▶ Nicht einfach mit `ret`
- ▶ Nicht einfach “ohne weitere Instruktion”
- ▶ Stattdessen: System Call `_exit` (60) oder `_exit_group` (231)
 - ▶ Jeweils 1 Parameter: Exit Code zw. 0–255
 - ▶ Kein Rückgabewert