

Übungen zu Systemnahe Programmierung in C (SPiC)

Peter Wägemann, Heiko Janker, Moritz Strübe, Rainer Müller
(Lehrstuhl Informatik 4)



Wintersemester 2014/2015



Inhalt

Aufgabe: tbsh

Theorieüberblick



Aufgabe: tbsh

Theorieüberblick



Aufgabe: tbsh – 4 Konzepte I

1. Signalhandler installieren (sigaction()):

```
1 struct sigaction act;
2 act.sa_handler = SIG_DFL; // Handlersignatur: void f(int signum)
3 act.sa_flags = SA_RESTART;
4 sigemptyset(&act.sa_mask);
5 sigaction(SIGINT, &act, NULL);
```

2. Signale blockieren/deblockieren (sigprocmask()):

```
1 sigset_t set;
2 sigemptyset(&set);
3 sigaddset(&set, SIGUSR1);
4 sigprocmask(SIG_BLOCK, &set, NULL); /* Blockiert SIGUSR1 */
5 // kritischer Abschnitt
6 sigprocmask(SIG_UNBLOCK, &set, NULL); /* Deblockiert SIGUSR1 */
```



Aufgabe: tbsh – 4 Konzepte II

3. Neuen Prozess erstellen (fork())

```
1 pid_t p = fork();
2 switch(p) {
3     case -1: // Fehler - kein Kind
4         ...
5     case 0: // Kind
6         ...
7     default: // Vater
8         ...
9 }
```

4. Programm im aktuellen Prozess starten (exec())

```
1 char *args[4];
2 args[0] = "cp";
3 args[1] = "x.txt";
4 args[2] = "y.txt";
5 args[3] = NULL;
6 execvp(args[0], args);
```



Aufgabe: tbsh – Vererbung der Signalmaske

■ Prozessweite Signal-Maske wird über exec(3) vererbt

```
1 struct sigaction act;
2 switch(pid = fork()) {
3     case -1:
4         perror("fork");
5         exit(EXIT_FAILURE);
6     case 0:
7         act.sa_handler = SIG_DFL;
8         act.sa_flags = SA_RESTART;
9         sigemptyset(&act.sa_mask);
10        sigaction(SIGINT, &act, NULL);
11
12        execvp(argv[0], argv);
13        perror(argv[0]);
14        exit(EXIT_FAILURE);
15    }
16    // Vater-Prozess läuft hier weiter ...
```

■ Installieren des Default-Signalhandlers vor dem exec(3)-Aufruf



https://www4.cs.fau.de/Lehre/SS14/V_SPIC/Folien/V_SPIC_handout.pdf

