Übungen zu Systemnahe Programmierung in C

Abschnitt 12.2: Minimale Shell

13.07.2020

Tim Rheinfels
Benedict Herzog
Bernhard Heinloth

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg

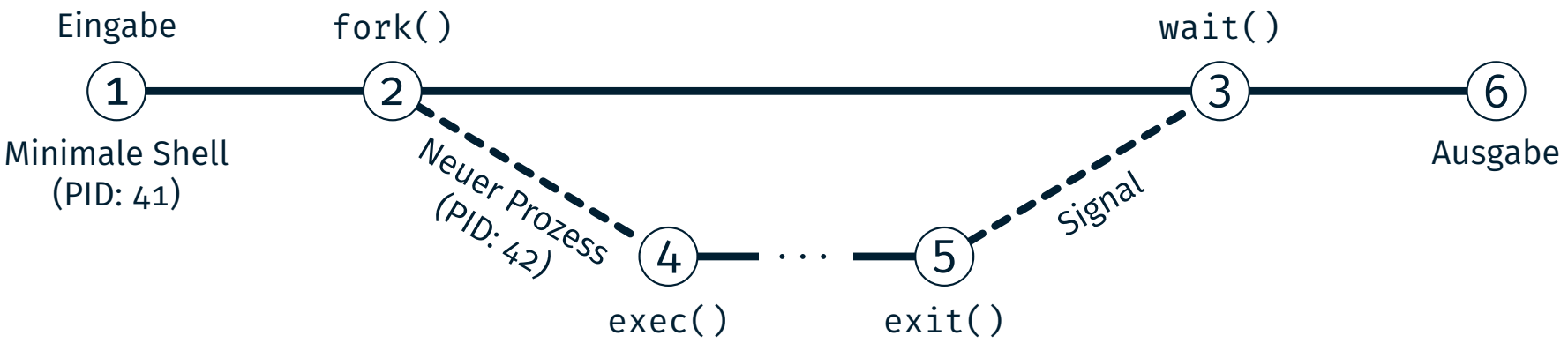
Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Funktionsweise einer minimalen Shell

1. Auf Eingaben vom Benutzer warten Prompt
2. Neuen Prozess erzeugen fork()
3. Vater: Wartet auf die Beendigung des Kindes wait()
4. Kind: Startet Programm exec()
5. Kind: Programm terminiert exit()
6. Vater: Ausgabe der Kindzustands



Einlesen von der Standard-Eingabe mit fgets

```
01 char *fgets(char *s, int size, FILE *stream);
```

- `fgets(3)` liest eine Zeile vom übergebenen Kanal
- `'\n'` wird mitgespeichert
- Maximal `size-1` Zeichen + finales `'\0'`
- Im Fehlerfall oder EOF wird NULL zurückgegeben

⇒ Unterscheidung `ferror(3)` oder `feof(3)`

```
01 char buf[23];
02 while (fgets(buf, 23, stdin) != NULL) {
03     // buf enthält Zeile
04 }
05
06 if(ferror(stdin)) {  // Fehler
07     [...]
08 }
```

↝ 11.3

Stringmanipulation mit `strtok()`

```
01 char *strtok(char *str, const char *delim);
```

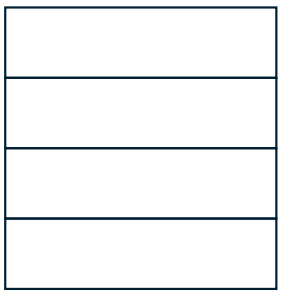
- `strtok(3)` teilt einen String in Tokens auf
- Tokens werden durch Trennzeichen getrennt
- Liefert bei jedem Aufruf Zeiger auf nächsten Token
- `delim:` String, der alle Trennzeichen enthält (z.B. `" \t\n"`)
- `str:`

  erster Aufruf Zeiger auf zu teilenden String

  alle Folgeaufrufe `NULL`

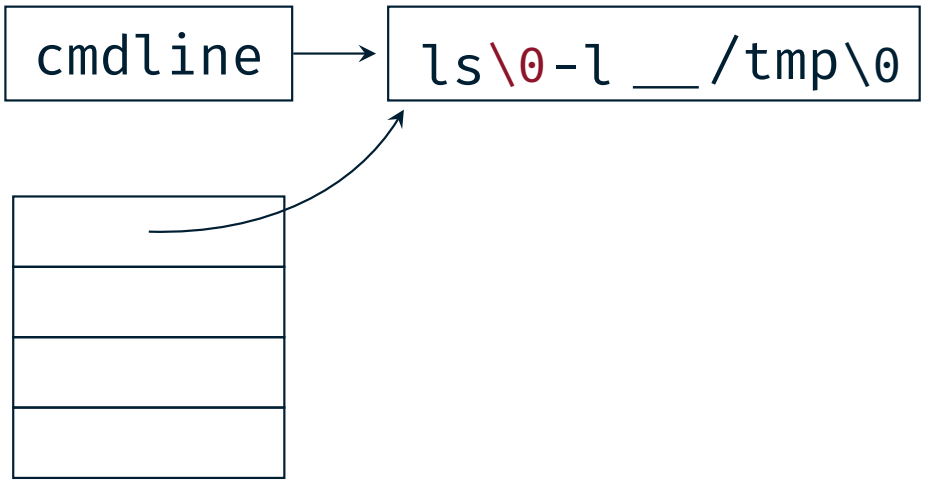
- Aufeinanderfolgende Trennzeichen werden übersprungen
- Trennzeichen nach Token werden durch `'\0'` ersetzt
- Am Ende des Strings: `strtok(3)` gibt `NULL` zurück

Stringmanipulation mit `strtok()`



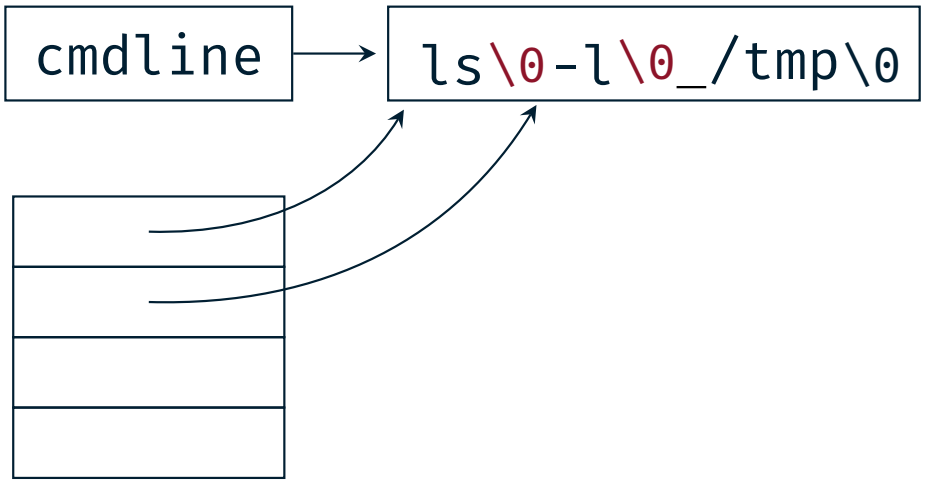
```
01 char cmdline[] = "ls -l  /tmp";
02 char *a[4];
03 a[0] = strtok(cmdline, " ");
04 a[1] = strtok(NULL, " ");
05 a[2] = strtok(NULL, " ");
06 a[3] = strtok(NULL, " ");
```

Stringmanipulation mit `strtok()`



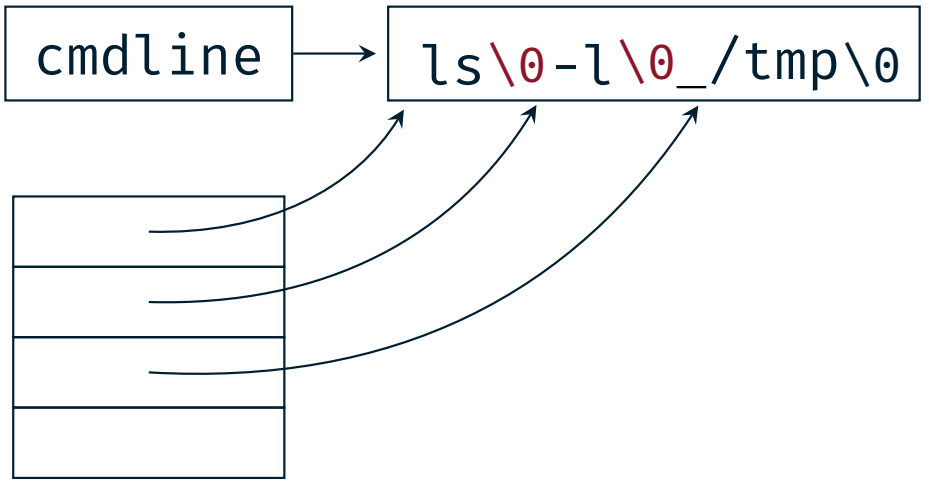

```
01 char cmdline[] = "ls -l  /tmp";
02 char *a[4];
03 a[0] = strtok(cmdline, " ");
04 a[1] = strtok(NULL, " ");
05 a[2] = strtok(NULL, " ");
06 a[3] = strtok(NULL, " ");
```

Stringmanipulation mit `strtok()`



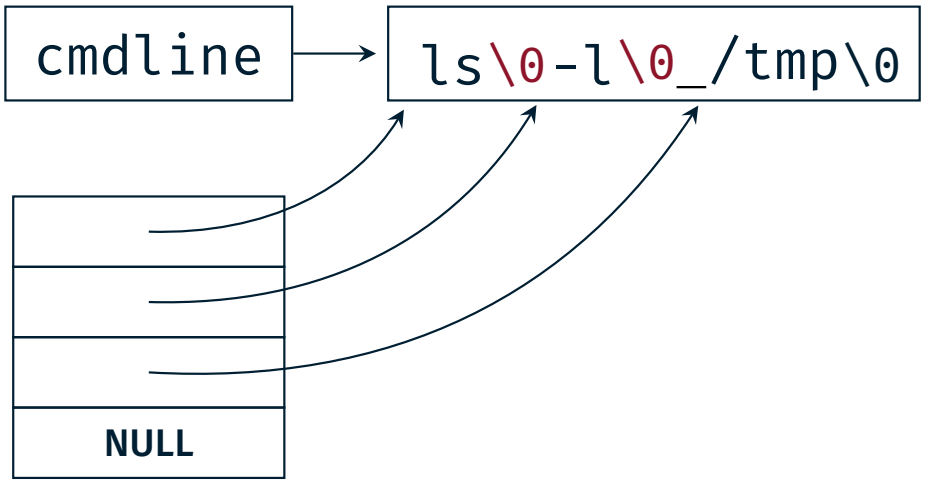

```
01 char cmdline[] = "ls -l  /tmp";
02 char *a[4];
03 a[0] = strtok(cmdline, " ");
04 a[1] = strtok(NULL, " ");
05 a[2] = strtok(NULL, " ");
06 a[3] = strtok(NULL, " ");
```

Stringmanipulation mit `strtok()`




```
01 char cmdline[] = "ls -l  /tmp";
02 char *a[4];
03 a[0] = strtok(cmdline, " ");
04 a[1] = strtok(NULL, " ");
05 a[2] = strtok(NULL, " ");
06 a[3] = strtok(NULL, " ");
```

Stringmanipulation mit `strtok()`




```
01 char cmdline[] = "ls -l  /tmp";
02 char *a[4];
03 a[0] = strtok(cmdline, " ");
04 a[1] = strtok(NULL, " ");
05 a[2] = strtok(NULL, " ");
06 a[3] = strtok(NULL, " ");
```