1 Variablen - Gültigkeitsbereich und Lebensdauer

- Lokale Variablen

```
void test() {
    int a;
    a = 42;
}
```

- Modul-lokale Variablen

```
static int a;
void test() {
    a = 42;
}
```

- Globale Variablen

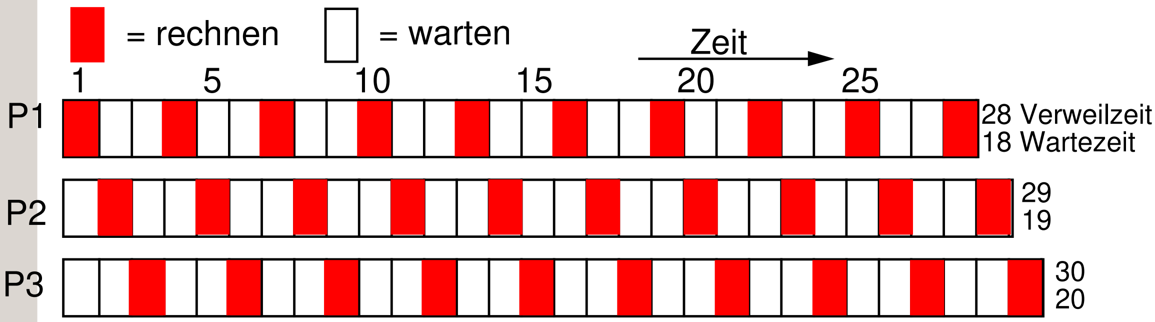
test.h
```
extern int a;
```

test.c
```
#include "test.h"
int a;
void test() {
    a = 42;
}
```

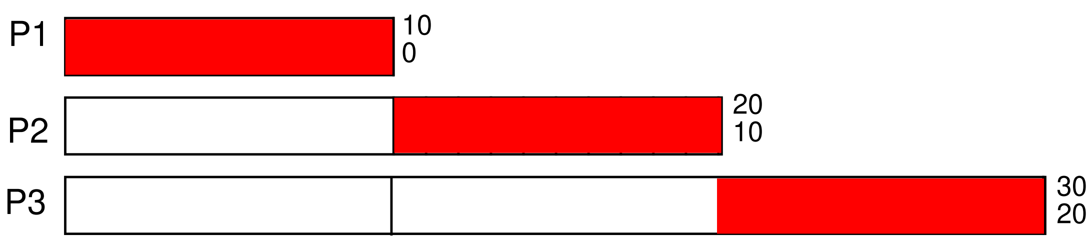
hallo.c
```
#include "test.h"

void hallo() {
    a = 13;
}
```


2 Zeitquantum und mittlere Verweilzeit

- 1 ms Quantum, 10ms Rechenzeit, Round Robin



- 10 ms Quantum, 10ms Rechenzeit, Round Robin

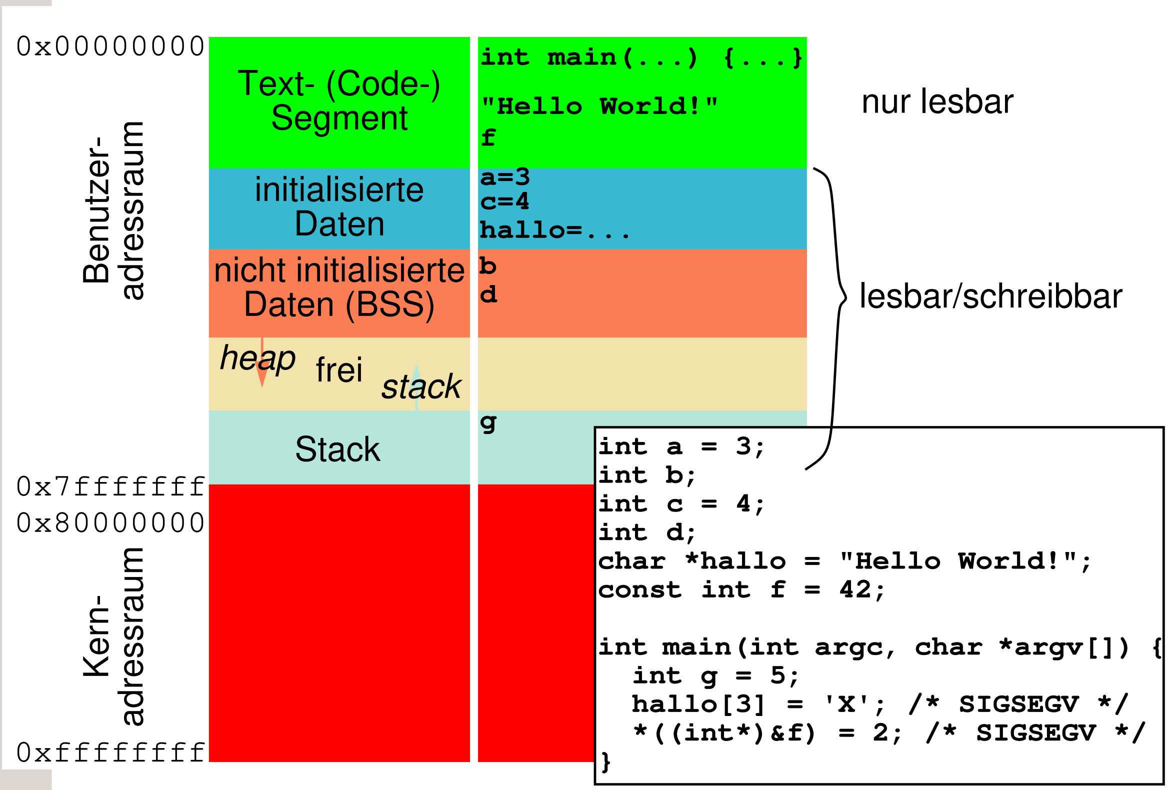


3 errno

- wird nach einem nicht erfolgreichen Systemaufruf gesetzt
- sofort nach Systemaufruf abfragen!
- Beispiel:

```
int fd;
fd = open("/etc/passwd", O_RDONLY);
if (fd == -1) {
    switch(errno) {
        case EACCES: printf("permission denied"); break;
        case EDQUOT: printf("quota limit reached"); break;
        case EEXIST: printf("file exists"); break;
        case ENOENT: printf("file not found"); break;
        ...
    }
} else {
    /* verwende fd */
    ...
}
```


4 Aufbau der Daten eines Prozesses



SP I
Übung zu Systemprogrammierung I
© Martin Steckermeier, Michael Golm, Universität Erlangen-Nürnberg, IMMD IV, 1997
uebung2.doc 1997-11-24 08.23
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 fork

- Vererbung von
  - Textsegment
  - Filedeskriptoren
  - Arbeitsverzeichnis
  - Benutzer- und Gruppen-ID (uid, gid)
  - Umgebungsvariablen
  - Signalbehandlung
  - ...
- Neu:
  - Prozess-ID
  - Datensegment, Stack
  - ...

6 exec

- Lädt Programm zur Ausführung in den aktuellen Prozeß
- ersetzt Text-, Daten- und Stacksegment
- behält: Filedeskriptoren, Arbeitsverzeichnis, ...
- Aufrufparameter:
  - Dateiname des neuen Programmes (z.B. **`"/usr/bin/cp"`**)
  - Argumente, die der **`main`**-Funktion des neuen Programms übergeben werden (z.B. **`"cp", /etc/passwd", "/tmp/passwd"`**)
  - evtl. Umgebungsvariablen
- Beispiel

```
execl("/usr/bin/cp", "cp", "/etc/passwd", "/tmp/passwd", 0);
```

7 exit

- beendet aktuellen Prozess
- gibt alle Ressourcen frei, die der Prozeß belegt hat, z.B.
  - Speicher
  - Filedeskriptoren (schließt alle offenen Files)
  - Kerndaten, die für die Prozeßverwaltung verwendet wurden
- Prozeß geht in den *Zombie*-Zustand über
  - ermöglicht es dem Vater auf den Tod des Kindes zu reagieren (wait)

8 wait

- warten auf Statusinformationen von Kind-Prozessen
  - **wait(int *status)**
  - **waitpid(pid_t pid, int *status, int options)**
- Beispiel:

```
int main(int argc, char *argv[]) {
  int pid;
  if ((pid=fork()) > 0) {
    /* parent */
    int status;
    wait(&status); /* Fehlerabfrage */
    printf("Kindstatus: %d", status);
  } else if (pid == 0) {
    /* child */
    execl("/usr/bin/cp", "cp", "/etc/passwd", "/tmp/passwd", 0);
    /* diese Stelle wird nur im Fehlerfall erreicht */
  } else { /* pid == -1 --> Fehler bei fork */ }
}
```