Betriebssysteme

Labor 9

Unix-Prozesse

- Jeder Prozess hat eine eindeutige Prozessidentifikator (eine positive ganze Zahl) = PID

Spezielle UNIX-Prozesse:

- der Prozess mit `PID=0` ist der Prozessplaner, der als **swapper** bezeichnet wird. Dieser Prozess ist Teil des Kernels und wird als Systemprozess betrachtet.
- der Prozess mit `PID=1` ist der Init-Prozess, der vom Kernel am Ende des Systemladevorgangs aufgerufen wird. Der Init-Prozess liest die systemabhängigen Initialisierungsdateien (/etc /rc* -Ressourcendateien) und bringt das System in einen stabilen Zustand (z. B. Mehrbenutzer). Der Init-Prozess ist ein Benutzerprozess, der mit Superuser-Rechten ausgeführt wird.

Unix-Prozesse

Erzeugen einen Prozess:

- wird mithilfe der Systemaufruf `fork()` durchgeführt.

```
#include <sys/types.h>
#include <unistd.h>
pid_t fork(void);
```

- Elternprozess (der Prozess, der die `fork()`-Funktion aufruft) vs. Kindprozess (der neu erstellte Prozess)
- **Rückgabe:**
  - Bei Erfolg:
    - `PID` des Kindprozesses im Elternprozess
    - `0` im Kindprozess
  - Bei Fehler: `-1`

Unix-Prozesse

Beendigung einen Prozess:

- Der Kindprozess signalisiert dem wartenden Elternprozess seinen Abschluss über den `exit()`-Systemaufruf. Der exit-Anruf überträgt über seinen Parameter eine Nummer, die dem Elternprozess eine normale oder fehlerhafte Beendigung signalisiert.
- **Rückgabe:**
  - `0` bei Erfolg
  - `!=0` bei Fehler
- Die `exit`-Syntax ist:

```
#include <stdlib.h>
void exit(int status);
```

- Der `exit`-Aufruf beendet den Prozess, indem er mit einem Statuscode ausgeführt wird, der dem höchstwertigen Byte des Statusworts `status` entspricht, und schließt alle von ihm geöffneten Dateien. Danach wird das `SIGCHLD`-Signal an den Elternprozess übertragen.

Unix-Prozesse

Zombie-Prozesse:

- In der `Unix`-Terminologie wird ein Prozess, der beendet wurde und für den der Elternprozess keine Wartezeit ausgeführt hat, als **`Zombie`** bezeichnet. In diesem Zustand hat der Prozess keine zugewiesenen Ressourcen, sondern nur seinen Eintrag in der Prozesstabelle. Der Kernel kann den gesamten vom Prozess verwendeten Speicher herunterladen und die geöffneten Dateien schließen.

- Wenn ein Prozess, dessen Elternprozess `init` ist, beendet wird, wird er kein `Zombie`, da der `init`-Prozess eine der `wait`-Funktionen aufruft, um den Status zu analysieren, in dem der Prozess beendet wurde. Durch dieses Verhalten vermeidet der `init`-Prozess das Laden des Systems mit `Zombie`-Prozessen.

Unix-Prozesse

Warten auf einen Prozess:

- Der Elternprozess kann mithilfe der `wait`- oder `waitpid`-Systemaufrufe auf den Abschluss (normal oder fehlerhaft) des Kindprozesses warten.
  Die `wait`/`waitpid`-Syntax ist:

```
#include <sys/types.h>
#include <sys/wait.h>
pid_t wait(int *status);
pid_t waitpid(pid_t pid, int *status, int options);
```

- **Rückgabe:**
  - `0` bei Erfolg
  - `−1` bei Fehler

- Das Argument `status` ist die Adresse des Statusworts.

Unix-Prozesse

Ein Prozess, der `wait` oder `waitpid` aufruft, kann:

- stecken bleiben (wenn alle seinen Kindprozessen ausgeführt werden);
- den Beendigungsstatus des Kindprozesses erhalten (wenn einer der Kindprozesse fertig ist);
- ein Fehler erhalten (wenn er keine Kindprozesse hat).

Unix-Prozesse

Die Unterschiede zwischen den beiden Funktionen sind:

1. `wait` blockiert den aufrufenden Prozess, bis ein Kindprozess beendet ist, während `waitpid` durch dem Argument `options` dies vermeidet;
2. `waitpid` wartet nicht auf den Abschluss des ersten Kindprozesses, sondern kann durch Argument `options` den erwarteten Kindprozess angeben;
3. `waitpid` ermöglicht die Steuerung von Programmen über Argument `options`.

- Wenn keine Kindprozesse vorhanden sind, gibt der `wait`-Aufruf den Wert `-1` zurück und setzt die Variable `errno` auf `ECHILD`.
- Die Art und Weise, wie der Kindprozess beendet wurde (normal oder fehlerhaft), kann mithilfe des `status`-Parameters ermittelt werden.

Unix-Prozesse

- Es gibt drei Möglichkeiten, einen Prozess abzuschließen: `exit`-Anruf, ein Beendigungssignal (`SIGKILL`) empfangen oder Systemabsturz. Der von der Variablen `status` zurückgegebene Statuscode gibt an, welcher der ersten beiden Modi die Beendigung verursacht hat (im dritten Modus verschwinden der Elternprozess und der Kernel, sodass `status` keine Rolle spielt).

Das `options`-Argument in der waitpid-Aufrufsyntax kann folgende Werte haben:

- `WNOHANG`: Der Anruf wird nicht blockiert, wenn das von der `PID` angegebene Kind nicht verfügbar ist. In diesem Fall ist der Rückgabewert `0`.
- `WUNTRACED`: Wenn die Implementierung die Steuerung der Arbeiten ermöglicht, wird der Status jedes gestoppten und nicht gemeldeten Prozesses zurückgegeben. `WIFSTOPPED` bestimmt, ob der zurückgegebene Wert einem gestoppten Kindprozess entspricht.

Das `options`-Argument kann auch 0 oder das Ergebnis einer der symbolischen Konstanten `WNOHANG` und `WUNTRACED` sein.

Unix-Prozesse

Abhängig vom `pid`-Wert lautet die Interpretation der `waitpid`-Funktion:

- `pid==-1`: das bedeutet, dass man auf einen Kindprozess warten muss;
- `pid>0`: dies bedeutet, dass man auf dem Kindprozess warten muss, dessen Prozess-ID dem Wert von `pid` entspricht;
- `pid==0`: dies bedeutet, dass man auf einen Kindprozess warten muss, dessen Prozessgruppen-ID der des aufrufenden Prozesses entspricht;
- `pid<-1`: dies bedeutet, dass man auf einen Kindprozess warten muss, dessen Prozessgruppen-ID dem absoluten Wert von `pid` entspricht.

Der waitpid-Aufruf gibt (`-1`) zurück, wenn es keinen Prozess oder eine Gruppe von Prozessen mit der angegebenen `pid` gibt oder diese `pid` keinem Kindprozess gehört.