Prof. Franz Schweiggert
Christian Ehrhardt
Abteilung Angewandte Informationsverarbeitung

30. Mai 2006
Blatt 6

Systemnahe Software (SS 2006)

Abgabetermin 13.Juni 2006

Signale zum zweiten (20 Punkte)

Die Signalverarbeitung muß ab sofort grundsätzlich mit Hilfe von `sigaction(2)` und nicht mit `signal(3C)` gemacht werden.

Parallelisierung Reloaded (16 Punkte)

In diesem Blatt soll ähnlich wie in Blatt 3 eine wiederholt zu erledigende mathematische Aufgabe auf eine fest vorgegebene Anzahl von Prozessen (z.B. 4) aufgeteilt werden. Dieses Mal sollen 64-Bit Zahlen (Typ `long long`) daraufhin untersucht werden, ob es sich um Primzahlen handelt oder nicht. Handelt es sich nicht um eine Primzahl soll eine Zerlegung in zwei Faktoren angegeben werden, ansonsten die Tatsache, daß es sich um eine Primzahl handelt. Die Eigentliche Zerlegung in Faktoren übernehmen die erzeugten Kindprozesse, der Vater kümmert sich nur um die Synchronisation. Er muß in Kooperation mit den Kindprozessen sicherstellen, daß jeder Eintrag in der Eingabedatei genau einmal verarbeitet wird. Dies soll wie folgt ablaufen:

- Alle Prozesse teilen sich eine einzige IO-Verbindung, das heißt, daß die Eingabedatei `INPUT` (Beispiele gibt es auf der Homepage der Vorlesung) nur ein einziges Mal vom Vater geöffnet wird.
- Die Kindprozesse lesen nur aus der Datei, wenn sie per Signal dazu aufgefordert wurden.
- Der Vater fordert zu jedem Zeitpunkt immer nur einen Kindprozeß zum Lesen aus der Datei auf und liest selbst nie. Dazu wird ein Signal (`DOREAD`) an den Kindprozeß versendet und der Vater wartet dann auf ein Antwortsignal das Kindprozesses.
- Der Kindprozeß beantwortet die Aufforderung zum Lesen aus der Datei mit
  - einem Signal (`BUSY`), das anzeigt, daß er noch mit einem früheren Auftrag beschäftigt ist oder
  - er ließt eine 64-Bit Zahl und bestätigt mit einem anderen Signal (`READDONE`, daß er eine Zahl gelesen hat und sie anschließend faktorisieren wird oder
  - der Kindprozeß versucht zu lesen, stellt fest, daß das Dateiende erreicht ist und teilt dies dem Vater mit Hilfe eines dafür vorgesehenen dritten Signals (`THEEND`) mit. Anschließend wird noch das Signal `BUSY` hinterher geschickt, das normalerweise anzeigt, daß der Kindprozeß noch mit einem früheren Auftrag beschäftigt ist.
- Das Ergebnis der Faktorisierung wird vom Kindprozeß auf die Standardausgabe ausgegeben. Nach dem Ende der Faktorisierung teilt der Kindprozeß dem Vater wieder durch ein dafür vorgesehenes Signal (`DONE`) mit, daß er seine Aufgabe erledigt hat. Und wartet auf eine erneute Aufforderung zum Lesen.
- Nach der Aufforderung an einen Kindprozeß zu lesen wartet der Vater immer auf ein Antwortsignal des Kindprozesses. Dieses zeigt an, ob der Kindprozeß noch beschäftigt ist.
- Sind im Moment alle Kindprozesse beschäftigt, legt sich der Vater für eine Sekunde schlafen und fordert dann erneut, die Kindprozesse zum Lesen auf. Sollte wärend der Vater schläft einer der Kindprozesse seinen Auftrag beenden, wird der Vater durch das dann versendete Signal `DONE` automatisch geweckt und schläft in diesem Fall kürzer als eine Sekunde.
- Stellt der Vater fest, daß das Dateiende erreicht ist, verschickt er ein `SIGTERM` Signal an alle erzeugten Kindprozesse und wartet auf deren Terminierung bevor er sich selbst beendet.
- Die Kindprozesse beenden beim Eintreffen eines `SIGTERM` Signals zuerst noch die aktuelle Faktorisierung und beenden sich dann.

Für die einzelnen Signale empfehle ich folgende Definitionen:

#define	DOREAD	(SIGRTMIN)	// Aufforderung an ein Kind zum Lesen
#define	READDONE	(SIGRTMIN+1)	// Kind hat erfolgreich gelesen
#define	BUSY	(SIGRTMIN+2)	// Kind ist noch beschäftigt
#define	DONE	(SIGRTMIN+3)	// Kind hat eine Faktorisierung beendet
#define	THEEND	(SIGRTMIN+4)	// Dateiende erreicht

Mehr Sicherheit (4 Punkte)

Das oben beschriebene Verfahren zur Synchronisation mit Hilfe von Signalen kommt völlig aus dem Tritt, wenn einer der beteiligten Prozesse versehentlich oder böswillig von einem unbeteiligten Prozeß eines der zur Synchronisation verwendeten Signale geschickt bekommt. Es soll daher jetzt sichergestellt werden, daß die Signale `DOREAD`, `READDONE`, `BUSY`, `DONE` und `THEEND` nur dann verarbeitet werden, wenn sie von einem der legitimen Absender stammen. Das heißt, die Kinder akzeptieren nur Signale vom gemeinsamen Vater, der Vater nur Signale von einem seiner Kinder.
Dazu muß bei `sigaction(2)` das Feld `sa_sigaction` statt `sa_handler` für die Signalverarbeitung verwendet werden. Außerdem muß `SA_SIGINFO` in `sa_flags` angegeben werden. Dann bekommt der Signalhandler neben der Signalnummer zwei zusätzliche Parameter, von denen uns nur der erste (ein Zeiger auf eine Struktur vom Typ `siginfo_t`) interessiert. Dieser Struktur lassen sich weitere Informationen über den Ursprung bzw. den Grund des Signals entnehmen.

Uns interessieren hierbei im Kontext dieser Aufgabe nur nutzergenerierte Signale und bei diesen nur die Prozeß-ID des Absenders. Nähere Informationen zu dieser Datenstruktur müssen der Manualseite `siginfo(3HEAD)` (bzw. `sigaction(2)` unter Linux) entnommen werden.

Viel Erfolg!