Aufgabe 7:

11.06.2007

job_sh (12 Punkte) Bearbeitung in Zweier-Gruppen

Programmieren Sie basierend auf der mini_sh von Aufgabe 3 eine um Hintergrundprozesse, Signalbehandlung und Job-Verwaltung erweiterte Shell: **job_sh** (**job sh**ell). Verwenden Sie als Ausgangsbasis dafür bitte unbedingt die Vorgabe **/proj/i4sos/pub/aufgabe7/mini_sh.c**. Die unten gestellten Fragen sind in der Dokumentation (**job_sh.txt**) zu erläutern.

a) Promptsymbol (RCS Release # 1)

Ändern Sie das Promptsymbol der Shell auf "job_sh>" und passen Sie das Makefile an die neue Aufgabe an. Legen Sie ein Unterverzeichnis für RCS an und checken Sie diese Version von **job_sh.c** als Version 1 ein (**ci -u1 job_sh.c**). Am Ende jeder weiteren Teilaufgabe soll nun das in Klammern angegebene Release eingecheckt werden. Pro Release können beliebig viele Level eingecheckt werden - der jeweils aktuellste Level jedes Release repräsentiert die Abgabe der jeweiligen Teilaufgabe.

b) Hintergrundprozesse (RCS Release # 2)

Wenn eine Kommandozeile mit dem Zeichen **&** abgeschlossen wird, soll das Kommando ähnlich wie bei einer UNIX-Shell als *Hintergrundprozess* ausgeführt werden. Die Shell soll nicht auf das Terminieren dieses Hintergrundprozesses warten, sondern sofort wieder das Promptsymbol ausgeben und das nächste Kommando von der Standardeingabe einlesen.

c) Warten auf Kindprozesse (RCS Release # 3)

Was passiert, wenn Hintergrundprozesse und ein Vordergrundprozess laufen und ein Hintergrundprozess zuerst fertig wird (☞ Dokumentation)? Stellen Sie in Ihrem Programm sicher, dass garantiert immer der Exit-Status des Vordergrundprozesses ausgegeben wird. Zum Testen können Sie das Programm **sleep(1)** als Hintergrund- bzw. Vordergrundprozess verwenden.

d) Signalhandler (RCS Release # 4)

Der shell-Prozess soll nun das Interrupt-Signal vom Terminal abfangen (**CTRL+C**). In der Signalbehandlungsfunktion soll nur die Meldung "Interrupt!" auf dem Standardfehlerkanal ausgegeben werden (**sigaction(2)**). Was passiert, wenn Ihr shell-Programm ein Interrupt-Signal erhält und nur ein Vordergrundprozess läuft bzw. wenn auch Hintergrundprozesse laufen (☞ Dokumentation)? Ändern Sie das Programm nun so, dass die Hintergrundprozesse das Signal **SIGINT** ignorieren. Was hat sich dadurch am Verhalten bei einem **CTRL+C** an das Terminal geändert (☞ Dokumentation)?

e) Jobverwaltung (RCS Release # 5)

Implementieren Sie in der Shell ein Kommando **jobs**, das die Kommandozeilen und Prozess-Ids aller laufenden Hintergrundprozesse ausgibt. Stellen Sie sicher, dass Ihre Jobliste immer aktuelle Informationen enthält, d.h. wenn ein Hintergrundprozess terminiert, soll er auch *unmittelbar* aus der Jobliste ausgetragen werden. Richten Sie hierfür einen Signalhandler für **SIGCHLD** ein. Geben Sie beim Terminieren eines Hintergrundprozesses auch dessen Exit-Status zusammen mit der Kommandozeile aus. Zur Verwaltung der Hintergrundprozesse (Jobs) verwenden Sie bitte die Implementierung aus **/proj/i4sos/pub/aufgabe7/joblist.[ch]** (Makefile anpassen nicht vergessen!).

f) Nebenläufigkeitsprobleme (RCS Release #6 oder bereits in früheren Releases enthalten)

Durch die potentiell nebenläufige Arbeit auf der Jobliste durch Signalbehandlungen und den eigentlichen Programmablauf kann es zu sog. *Race Conditions* kommen. Identifizieren Sie die möglichen Probleme und beschreiben Sie diese zusammen mit Ihren Lösungen zur Vermeidung dieser Probleme in der Dokumentation (**sigprocmask(2)**).

Abgabe: bis spätestens Donnerstag, 21.06.2007, 12:00 Uhr

Bei Fragen zur Aufgabenstellung bitte Mail an *i4sos@informatik.uni-erlangen.de*