Aufgabe 2: myfind mit exec (12 Punkte, Abgabe Fr, 6.6.2008 20:00)

Erweitern Sie das Programm **myfind** aus Aufgabe 1 um folgende Optionen:

```
myfind path... [-exec command {} ;] [-maxproc n]
```

In */proj/i4sos/pub/aufgabe2* finden Sie eine Musterlösung zu Aufgabe 1 sowie weitere Vorgaben, auf denen Sie aufbauen können.

a) Ausführen von Kommandos (`-exec command {} ;`)

Ist die Option `-exec` angegeben, so soll anstelle der Ausgabe von Datei- und Verzeichnisnamen ein Kommando für jeden Verzeichniseintrag, der alle Suchkriterien erfüllt, ausgeführt werden. Das Kommando kann Optionen erhalten, die durch einen **';'** abgeschlossen werden. Außerdem wird der String **'{}'**, welcher als *eigenes Argument* maximal *einmal* im Kommando vorkommen kann, durch den jeweiligen Dateinamen ersetzt. Um eine Interpretation durch die Shell zu verhindern kann es sowohl für die geschweiften Klammern als auch für den Strichpunkt notwendig sein, diesen die Sonderbedeutung zu nehmen, beispielsweise durch Voranstellung eines Backslash **'\'**.

Zur Ausführung eines Kommandos müssen Sie einen Kindprozess für jeden die Suchkriterien erfüllenden Dateinamen erzeugen (**fork(2)**) und anschließend das Kommando ausführen (**execvp(3)**). Das Kommando liegt nach dem Aufruf der vorgegebenen Funktion **parse_command()** bereits in der von **execvp** benötigten Form im Argumentenvektor vor. Richten Sie zur *unmittelbaren* Beseitigung von Zombieprozessen eine Signalbehandlungsroutine ein.

b) Begrenzung der maximalen Prozesszahl

Mit der `-maxproc` Option soll nun zusätzlich die Anzahl der laufenden Kindprozesse begrenzt werden können. Ist die Option nicht angegeben, so soll die Zahl auf einen Prozess begrenzt sein. Ist die maximale Prozesszahl erreicht soll das Programm auf die Beendigung eines Kindprozesses warten (**sigsuspend(2)**). Danach kann ein neuer Prozess erzeugt werden. Achten Sie auf eine korrekte Synchronisation von Signalbehandlung und Hauptprogramm um eine Fehlfunktion Ihres Programmes in Folge einer Race Condition zu vermeiden (**sigprocmask(2)**).

c) Ausgabe des Exitstatus

Ergänzen Sie Ihr Programm nun um die Ausgabe des Exitstatus für jeden erzeugten Kindprozess. Verwalten Sie die laufenden Kindprozesse hierfür in einer Jobliste, welche in der Datei **joblist.o** vorgegeben wird. Beim Terminieren eines Kindprozesses sollen dessen Kommandozeile und Exitstatus unmittelbar ausgegeben und der Job aus der Jobliste entfernt werden:

```
[command ... filename] with pid [pid] terminated with status [status]
```

Der Status zeigt an, ob das Kommando sich normal beendet hat oder durch ein Signal terminiert wurde. Hier ein Beispiel für die Ausgabe einer Datei mittels

`myfind /proj/i4sos/pub -exec ls -l {};`:

```
[ls -l aufgabe1] with pid [1234] terminated with status [Exit 0]
[ls -l aufgabe2] with pid [1235] terminated with status [Signal 3]
```

Achten Sie bei der Manipulation der Liste auf Nebenläufigkeitsprobleme, die durch das parallele Ausführen von Signalhandler und normalen Programmfluss entstehen!

Die Aufgabe kann in 2er-Gruppen bearbeitet werden. Verwenden Sie die vorgegebene Funktion **parse_command()**, die bereits alle notwendigen Kommandozeilenparameter parst und den Argumentenvektor in eine leicht weiterverarbeitbare Form bringt. Die Funktion ist in der zugehörigen Headerdatei dokumentiert. Es steht ausserdem ein Makefile zur Verfügung.

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