ulm university universität
uulm

Institut für Numerische Mathematik

Dr. Andreas F. Borchert

7. Mai 2019
Blatt 2

Systemnahe Software II (SS 2019)

Abgabe bis zum 14. Mai 2019, 14:00 Uhr

Lernziele:

- Zusammenspiel von fork(), exec(), exit() und wait().

Aufgabe 2: Kommando parallel für eine Serie von Dateien ausführen

Schreiben Sie ein Werkzeug namens *par*, das ein Kommando für eine Reihe von Dateien per *fork()* und *exec()* parallel ausführen soll. Folgende Paramater sollen als Kommandozeilenargumente beim Aufruf übergeben werden:

- Der Name des Kommandos, das ausgeführt werden soll, mit seinen weiteren Kommandozeilenargumenten. Da die Zahl der Argumente variabel ist, wird das gesamte Kommandos in frei wählbare Trennzeichen eingefasst.
- Muster der Eingabedateien. Ein Muster muss genau ein „%“ enthalten und alle Eingabedateien müssen dem Muster entsprechen, wobei „%“ dann einer beliebigen nichtleeren Zeichensequenz entspricht. Beispiel: Bei dem Muster „%.c“ und dem Dateinamen „main.c“ entspricht das „%“ dem „main“.
- Muster der Ausgabedateien, die ebenfalls genau ein „%“ enthalten. Hierbei wird das „%“ durch die Zeichensequenz ersetzt, die beim Eingabe-Muster durch das „%“ abgedeckt wurde. Wenn passend zum obigen Beispiel als Muster „%.i“ angegeben wird, dann wird der Eingabedatei „main.c“ die Ausgabedatei „main.i“ zugeordnet.
- Beliebige viele Eingabedateien, die dem Muster der Eingabedateien entsprechen müssen.

Mit dem folgenden Kommando wird für jede in „.c“ endende Datei der C-Präprozessor *cpp* so aufgerufen, dass die Standard-Eingabe mit der in „.c“ endenden Datei verbunden ist und die Standard-Ausgabe mit der zugehörigen „.i“-Datei.

```
theon$ par X cpp -std=c11 X %.c %.i *.c
theon$
```

All die Kommandoaufrufe erfolgen dabei parallel und *par* terminiert genau dann, wenn all die aufgerufenen Kommandos beendet sind. Nur wenn diese alle erfolgreich enden, sollte mit einem Exit-Code von 0 terminiert werden. Ansonsten sind diagnostische Fehlermeldungen auszugeben, wobei die jeweilige Eingabedatei zu benennen ist.

Wenn beim Aufruf nicht genügend Argumente übergeben werden, soll das Programm eine kurze Usage-Meldung ausgeben:

```
theon$ par
Usage: par delimiter command ... delimiter in-pattern out-pattern {file}
theon$
```

Hinweise:

Alle Fehlermeldungen von *par* sind auf der Standard-Fehlerausgabe (Dateideskriptor 2) auszugeben. Mit dem Systemaufruf *dup2* lässt sich ein Dateideskriptor auf einen gegebenen anderen Deskriptor duplizieren, die dann beide auf den gleichen Eintrag in die *open file table* verweisen. Auf diese Weise lassen sich neue Datei-Verbindungen mit vorgegebenen Dateiskriptoren (z.B. 0 für die Standard-Eingabe, 1 für die Standard-Ausgabe) verknüpfen. Nach dem Aufruf von *dup2* sollte der alte Dateideskriptor geschlossen werden.
Ihre Lösung können Sie dann mit *submit* einreichen:

```
submit ss2 2 team [notes] par.c
```

Viel Erfolg!