OPERATING SYSTEMS BEISPIEL 2

Aufgabenstellung – forksort

Schreiben Sie ein Programm, das Eingaben sortiert.

```
SYNOPSIS
    forksort
```

Anleitung

Das Programm soll die Daten von *stdin* lesen. Die erste Zeile muss eine Zahl größer Null sein, die die Anzahl der zu sortierenden Strings angibt. Die darauffolgenden Zeilen enthalten die Strings selbst; Sie können die Strings auf je 62 echte Zeichen begrenzen. (Vergessen Sie nicht dafür eine Konstante zu definieren!)

Arbeitsweise

Forksort funktioniert wie eine rekursive Merge-Sort-Variante[1].

1. Lesen Sie die Anzahl der Strings ein.
2. Ist die Anzahl gleich Eins, dann geben Sie das „sortierte“ Ergebnis aus (d.h. den Eingabe-String selbst).
3. Ist die Anzahl größer Eins, erstellen Sie zuerst vier Unnamed-Pipes (zwei je Kind) und erzeugen dann zwei Kindprozesse (nicht Kind- und Enkelkind) mittels *fork(2)*. Verwenden Sie *dup2(2)* und *execlp(3)* um *stdin*/*stdout* der neuen Prozesse auf eigene Streams umzuleiten und danach Forksort rekursiv aufzurufen.
4. Schreiben Sie je die Hälfte der Strings auf *stdin* der beiden Kinder im zuvor spezifiziertem Eingabeformat (d.h., Anzahl der Strings, Newline, die Eingabestrings zeilenweise). Ist die Anzahl der Strings ungerade, dann wählen Sie ein beliebiges Kind aus, das einen String mehr bekommt.
5. Lesen Sie nun die Ausgabe der Kinder zeilenweise ein, und geben Sie diese aber nun ”verschmolzen”, d.h. sortiert zeilenweise aus (dieser Vorgang nennt sich *Merge*). Beachten Sie, dass die Ausgabe der Kinder bereits fuer sich sortiert ist. Sie können also solange die Strings des ersten Kindes ausgeben, bis der kleinste String des zweiten Kindes kleiner als der kleinste, noch nicht ausgegebene, String des ersten Kindes ist. Danach lesen Sie vom zweiten Kind, bis der String des ersten Kindes wiederum kleiner ist. Wiederholen Sie diesen Vorgang solange, bis alle Strings der Kinder abgearbeitet sind.

   Der Merge-Vorgang muss einen linearen Aufwand haben.
6. Verwenden Sie *wait(2)* oder *waitpid(2)*, um den Exit-Status der beiden Kinder zu lesen.

Hinweise

Achten Sie auf korrekte Terminierungsbedingungen von forksort um ”Endlos-Rekursionen” zu vermeiden[2]. Dazu sind zwei Regeln zu beachten:

[1] Siehe insb. die graphisch aufbereitete Beispielausführung auf: http://de.wikipedia.org/wiki/Mergesort
[2] http://en.wikipedia.org/wiki/Fork_bomb

1. Forken Sie nur, wenn die Anzahl der zu sortierenden Strings größer eins ist.
2. Die an das jeweilige Kind übergebene Anzahl muss stets kleiner sein als die des Elternprozesses.

Beginnen Sie am besten, indem Sie eine Variante schreiben, die nur einen String sortieren kann. Erweitern Sie nun Ihr Programm für zwei Strings, indem Sie den beschriebenen Fork-Vorgang implementieren und jeweils einen String an jedes der beiden Kinder schreiben. Diese können ja schon einen einzelnen String sortieren. Ein einziger Aufruf von *strcmp(3)* als Merge-Implementation genügt um zu entscheiden welche Ausgabe der Kinder zuerst vom Elternprozess ausgegeben werden soll. Wenn das funktioniert, können Sie Ihre Merge-Implementierung verallgemeinern, um auch mehr als zwei Strings sortieren zu können.

Um Fehlermeldungen und Debug-Messages auszugeben, verwenden Sie stets *stderr*, da die *stdout* in den meisten Fällen umgeleitet ist.

Zum Testen der Merge-Implementierung können Sie mit *execlp(3)* statt Forksort auch `sort(1)` aufrufen. Für die endgültige Abgabe ist diese Vorgehensweise nicht gültig, zum Testen aber durchaus sinnvoll.

Richtlinien

Bitte beachten Sie auch die *Richtlinien für die Erstellung von C-Programmen* sowie die *Allgemeinen Hinweise zur Beispielgruppe 2* auf der Übungswebsite.

Insbesondere ist es ab dieser Beispielgruppe notwendig, die Dokumentation in Doxygen zu führen. Es muss zumindest das HTML-Output generierbar sein. Bitte dokumentieren Sie ausnahmslos alle Funktionen (auch `static`-Funktionen; siehe `EXTRACT_STATIC` in der `Doxyfile`). Eine kurze Einführung haben wir Ihnen auf `http://wiki.vmars.tuwien.ac.at/index.php/Doxygen_Primer` bereitgestellt. Achten Sie weiters darauf, dass nach außen hin sichtbare Funktionen (exportierte Funktionen) in der Header-Datei und lokale (`static`) Funktionen nur in der C-Datei dokumentiert werden. Sie sollten auch Ihre Typen (insbesondere `structs`), Konstanten und globale Variablen dokumentieren.