OPERATING SYSTEMS BEISPIEL 2

Aufgabenstellung – stillepost

Gründen Sie eine Prozessfamilie bestehend aus einem Elternprozess und n Kindprozessen (nicht Kind und Enkelkind, $n \geq 2$). Spielen Sie dann mit der Familie ein Stille-Post-Spiel.

Das von Ihnen zu entwickelnde Programm `stillepost` soll sich folgendermaßen verhalten:

- Der Benutzer des Programms überlegt sich einen beliebigen String.
- Er ruft das Programm (also den Elternprozess) mit dem String als Parameter auf. Beispiel: `stillepost 25 "OSUE macht Spass."`
- Der Elternprozess sagt den String an sein erstes Kind weiter, jedoch nicht ohne eine kleine Veränderung einzubauen – wie beim Stille-Post-Spiel üblich.
- Das erste Kind erhält diesen String, verändert ihn wieder geringfügig und gibt ihn an das zweite Kind weiter.
- Das zweite Kind verfährt ebenso wie das erste, nur gibt es den von ihm veränderten String nicht an einen anderen Prozess weiter, sondern sagt ihn laut (gibt ihn auf `stdout` aus).

Anleitung

Das Programm soll folgende Synopsis befolgen:

```
SYNOPSIS:
    stillepost [-v] <n> <string>
```

Beachten Sie: Wenn Sie einen String übergeben wollen, der Leerzeichen enthält, müssen Sie ihn in Hochkommata einschließen, damit er als nur ein Argument behandelt wird: `"String mit Leerzeichen"`

Der Elternprozess soll seine Kinder mittels *fork(2)* erzeugen, die Kommunikation zwischen den Prozessen soll über Unnamed Pipes erfolgen, welche mit *pipe(2)* erzeugt werden.

Die Veränderungen eines jeden Mitspielers (Prozesses) sollen so vorgenommen werden, dass mittels *rand(3)* eine zufällige Position innerhalb des Strings gewählt, und dann das Zeichen an dieser Stelle durch ein anderes, ebenfalls zufälliges Zeichen aus `[a-zA-Z]` ersetzt wird.

Stellen Sie mittels *srand(3)* sicher, dass alle Prozesse eine unterschiedliche Random-Seed verwenden, damit nicht jeder Prozess die selben Änderungen vornimmt.

Die Option `-v` schließlich soll es ermöglichen, das Spiel aus der Sicht eines allwissenden Betrachters (bzw. jemand mit ausgezeichnetem Gehöhr) zu verfolgen. Wird die Option angegeben, gibt jeder Prozess den String zuerst unverändert und dann in der veränderten Form (wie er ihn weitergeben wird) auf *stdout* aus.

Führen Sie eine sinnvolle Beschränkung für die Länge des Strings ein.

Testen

Testen Sie ihr Programm vor allem mit der Option `-v`, da dann deutlich sichtbarer ist, was jeder einzelne Prozess tut.

Beispielrunde:

```
$ ./stillepost -v 2 "OSUE macht Spass."
Parent: erhalten: OSUE macht Spass.
Parent: weiter  : OSUE macht Xpass.
child1 : erhalten: OSUE macht Xpass.
child1 : weiter  : OSkE macht Xpass.
child2 : erhalten: OSkE macht Xpass.
child2 : weiter  : OSkE macAt Xpass.
child2 : Ende    : OSkE macAt Xpass.
```

Die Ausgabe ihres Programms muss nicht genau wie in dem Beispiel formatiert sein. Natürlich wird der String umso unkenntlicher, je kürzer er ist.

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 `struct`s), Konstanten und globale Variablen dokumentieren.