OPERATING SYSTEMS BEISPIEL 2

Aufgabenstellung – randsched

Sie sind der neue Systemadministrator im lokalen Atomkraftwerk. Schon bald ist Ihnen klar, dass im Wesentlichen einfach nur alle paar Minuten ein Programm aufgerufen werden muss, dessen Output in eine Protokolldatei abzutippen ist. Falls doch mal ein Fehler auftritt, muss aber von Ihnen der sofortige Reaktor-Shutdown eingeleitet werden. Sich Ihrer kritischen, dennoch gefährlichen und schlecht bezahlten Position bewusst, fassen Sie schnell den Entschluss, diesen Vorgang mit einem einfachen Scheduler Programm zu automatisieren. Damit diese Automatisierung nicht gleich Ihren Arbeitsgebern auffällt, soll Ihr Scheduler ebenfalls eine unregelmäßige Ausführung im Sinne von 'nur alle paar Minuten' unterstützen.

Anleitung

Implementieren Sie die folgenden Hilfsprogramme, um den Reaktor zu simulieren:

```
SYNOPSIS:
    rventgas

SYNOPSIS:
    rshutdown
```

Das Programm `rventgas` soll mit einer Wahrscheinlichkeit von 6/7 erfolgreich mit der Meldung *STATUS OK* terminieren, sonst soll *PRESSURE TOO HIGH - IMMEDIATE SHUTDOWN REQUIRED* ausgegeben werden und die Terminierung erfolgt mit einem Fehlercode.

Das Programm `rshutdown` schreibt mit einer Wahrscheinlichkeit von 1/3 die Meldung *SHUTDOWN COMPLETED* auf *stdout* und terminiert, sonst soll *KaBOOM!* ausgegeben und ein Fehlercode zurückgegeben werden.

Nun zum eigentlichen Teil der Aufgabe: Implementieren Sie das folgende Programm:

```
SYNOPSIS:
    schedule [-s <seconds>] [-f <seconds>] <program> <emergency> <logfile>

    -s <seconds>   Zeitfenster Anfang (Default: 1 Sekunde)
    -f <seconds>   max. Zeitfenster Dauer (Default: 0 Sekunden)
    <program>      Programm inkl. Pfad, das wiederholt ausgefuehrt werden soll
    <emergency>    Programm inkl. Pfad, das im Fehlerfall ausgefuehrt wird
    <logfile>      Pfad zu einer Datei, in der die Ausgabe von <program> sowie
                   Erfolg/Misserfolg von <emergency> protokolliert werden
```

Das Programm `schedule` erzeugt einen Kindprozess, der wiederholt zufällig irgendwann im spezifizierten Zeitfenster (in s bis $s+f$ Sekunden, als Granularität genügt eine volle Sekunde) das Programm `program` als Enkelkind startet und auf seine Beendigung wartet. Falls `program` mit einem Fehlercode terminiert, wird einmalig `emergency` ausgeführt und der Kindprozess beendet. Wird z.B. `schedule` mit `-s 5` und `-f 2` aufgerufen, wartet der Kindprozess zunächst zwischen 5 und 7 Sekunden und führt dann `program` aus. Terminiert `program` fehlerfrei, wartet der Kindprozess wieder zwischen 5 und 7 Sekunden, bevor er `program` ein weiteres Mal ausführt.

Weiters soll der Kindprozess alle Daten, die der `program`-Enkelprozess auf *stdout* ausgegeben hat, über eine Pipe an den Elternprozess schicken. Dieser gibt das Gelesene auf (sein) *stdout* aus und fügt es gleichzeitig an das angegebene `logfile` an. Ausgaben von `emergency` dürfen nicht im Logfile aufscheinen; es genügt zu protokollieren, ob `emergency` ohne Fehlercode terminiert hat. Nachdem `emergency` beendet ist, soll auch `schedule` terminieren. Implementieren Sie außerdem eine Signalbehandlung, um `schedule` sauber beenden zu können, ohne auf eine Fehlfunktion des Reaktors hoffen zu müssen.

Testen

Das typische Output sollte so ähnlich aussehen:

```
$ ./schedule -s 2 -f 5 ./rventgas ./rshutdown logfile
STATUS OK
STATUS OK
STATUS OK
STATUS OK
STATUS OK
PRESSURE TOO HIGH - IMMEDIATE SHUTDOWN REQUIRED
SHUTDOWN COMPLETED.
EMERGENCY SUCCESSFUL

$ cat logfile
STATUS OK
STATUS OK
STATUS OK
STATUS OK
STATUS OK
PRESSURE TOO HIGH - IMMEDIATE SHUTDOWN REQUIRED
EMERGENCY SUCCESSFUL
```

Die Meldung *SHUTDOWN COMPLETED* ist eigentlich von `rshutdown` und scheint nicht in der Log-Datei auf, wo aber sehr wohl *EMERGENCY SUCCESSFUL* anzeigt, dass der `emergency`-Prozess erfolgreich war.

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.