Institut für Informatik, Arbeitsbereich PVS WS 22/23

Prof. Sergei Gorlatch

Übungen zu Betriebssysteme

Blatt 2 (Prozesse und Signalisierung)

Aufgabe 3 (Prozesse und Adressräume)

5 Punkte

Schauen Sie sich das auf der Webseite angegebene Programm an und übersetzen Sie es.

a) Führen Sie das Programm aus und vergleichen Sie die beiden ausgegebenen Speicheradressen. Was fällt Ihnen auf? Erklären Sie, wie Ihre Beobachtung zustande kommt!

b) Anschließend führen Sie das Programm mehrfach aus. Wie verhalten sich die ausgegebenen Speicheradressen der verschiedenen Ausführungen zueinander? Erklären Sie welche betriebssysteminterne Mechanismus dafür verantwortlich ist!

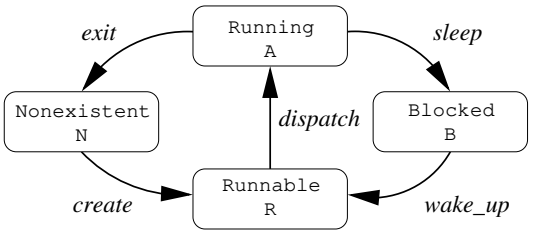
c) Führen Sie das Programm nun mehrfach mit `gdb` aus. Geben Sie dazu in der Konsole `gdb mmu_proof` und anschließend mehrfach `run` ein. Beschreiben Sie auch hier Ihre Beobachtungen bezüglich der Speicheradressen und vergeichen Sie sie mit der Beobachtung, die Sie in Aufgabenteil *b)* gemacht haben. Warum verhalten sich die Adressen anders, wenn das Programm mit `gdb` ausgeführt wird?

d) Führen Sie das Programm in zwei gleichzeitig geöffneten Terminal-Fenstern mit `gdb` aus. Starten Sie zunächst im linken Terminal das Programm mit "`run`", anschließend im rechten Terminal mit "`run` <ADDR>", wobei Sie für <ADDR> die Adresse des Stack-Speichers aus dem linken Terminal eingeben. Welches Verhalten würde man (naiv) erwarten? Warum tritt dieses Verhalten nicht ein?

Aufgabe 4 (Prozesszustände)

10 Punkte

In der Vorlesung (Kapitel 3) wurden verschiedene Zustandsgraphen für Prozesse vorgestellt. Für diese Übungsaufgabe sei folgender Zustandsgraph gegeben:



Gehen Sie davon aus, dass ein Prozess nicht verdrängt werden kann, sondern den Prozessor ausschließlich durch das Warten auf eine Bedingung oder bei Beendigung des Prozesses freigibt. Wenn der Prozessor frei ist und mehrere Prozesse lauffähig sind, so wird der Prozess mit der kleinsten Nummer ausgewählt.

Betrachten Sie nun den Ablauf der folgenden drei Prozesse, wobei zunächst nur Prozess P_3 existiert:

P_1:
11: erzeugt P_2
12: setzt B_1 auf wahr
13: wartet auf B_2
14: beendet sich

P_2:
21: setzt B_2 auf wahr
22: wartet auf B_4
23: beendet sich

P_3:
31: erzeugt P_1
32: erzeugt P_4
33: wartet auf B_2
34: wartet auf B_4
35: setzt B_3 auf wahr
36: beendet sich

P_4:
41: setzt B_4 auf wahr
42: wartet auf B_3
43: beendet sich

Stellen Sie die Zustände der Prozesse in einer Tabelle dar, die in etwa wie folgt aussieht:
P_1: N ...
P_2: N ...
P_3: A A ...
P_4: N ...

Aufgabe 5 (Signalisierung)

15 Punkte

Gegeben seien zwei Prozesse P_1 und P_2 und innerhalb dieser Prozesse jeweils ein Programmabschnitt A bzw. B. Mit Hilfe von Koordinationsoperationen soll eine bestimmte Reihenfolge der Bearbeitung erzwungen werden. Benutzen Sie die folgende Notation:

- A ist ausgeführt, bevor B beginnt: $A < B$
- Kompliziertere Zusammenhänge können durch logische Verknüpfung dargestellt werden, z. B. $A < B \vee B < A$

Programmtechnisch kann man eine solche Serialisierung durch die beiden Operationen `signal` (Signalisieren) und `swait` (Warten) auf einer binären Variable `s` erzwingen (Kapitel 4 der Vorlesung):

- `signal(s)` Setzt die Variable `s` (`s:=1`)
- `swait(s)` Falls `s` den Wert `1` hat, wird `s:=0` gesetzt und terminiert, andernfalls wird der aufrufende Prozess so lange blockiert, bis `s` den Wert `1` annimmt (d. h. ein anderer Prozess `signal` aufruft) und dann `s:=0` gesetzt und terminiert.

Betrachten Sie die folgenden Beispiele und geben Sie durch entsprechende Formel an, ob und wenn ja, welche Reihenfolgebeziehung jeweils zwischen A, B und C realisiert wird.

a) Initialisierung: `s:=0`
P_1: A `signal(s)`
P_2: `swait(s)` B

b) Initialisierung: `s1:=0 s2:=0`
P_1: `swait(s2)` A `signal(s1)`
P_2: `swait(s1)` B `signal(s2)`

c) Initialisierung: `s:=1`
P_1: `swait(s)` A `signal(s)`
P_2: `swait(s)` B `signal(s)`

d) Initialisierung: `s1:=0 s2:=0`
P_1: `signal(s1)` A `signal(s2)`
P_2: `swait(s1)` B `swait(s2)`

e) Initialisierung: `s1:=1 s2:=0`
P_1: `swait(s1)` A `signal(s2) signal(s1)`
P_2: `swait(s1)` B `signal(s2) signal(s1)`
P_3: `swait(s2)` C

f) Initialisierung: `s1:=0 s2:=0`
P_1: `signal(s1) swait(s2)` A
P_2: `signal(s2) swait(s1)` B

Hinweise zur Abgabe:

Das Übungsblatt muss bis zum **14.11.2022, 12:00 Uhr** abgegeben werden.
Halten Sie sich strikt an die Vorgaben im LearnWeb: *siehe hier*. Nichteinhalten der Vorgaben führt automatisch zu Punktabzug!

Die Bearbeitung muss in Gruppen von 3 oder 4 Teilnehmern erfolgen.
Fragen können in der Übung oder im LearnWeb geklärt werden. Abgabe per E-Mail an den jeweiligen Tutor der entsprechenden Übungsgruppe mit Subject „`Abgabe Uebung 2`“. Textaufgaben müssen als PDF-Datei abgegeben werden.

Bei der E-Mail Abgabe bitte nur **eine einzige** .zip oder .tgz oder .tbz Datei abgeben!

Die Abgaben sollen in den entspechenden Vorgabedateien implementiert werden. Die Abgaben müssen sich auf einem Linux-System der IVV5 mit dem bereitgestellten Makefile übersetzen lassen.

Wichtig: Bei der Abgabe in der E-Mail *alle* Namen und Matrikelnummern angeben.
Pro fehlender Angabe (Name oder Matrikelnummer) kann ein Punkt abgezogen werden!