Übungsblatt 3

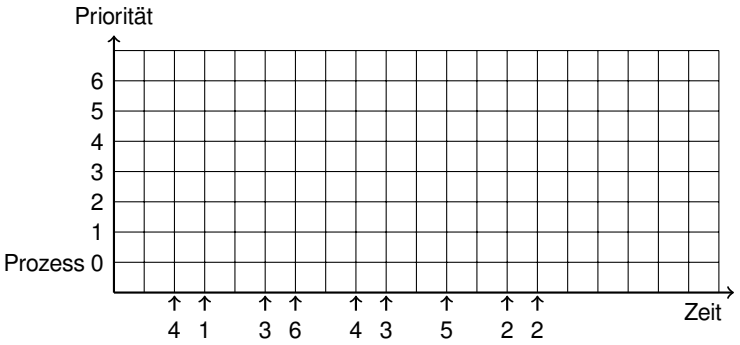
Abgabe bis spätestens 16.11.2023 in Stud.IP

Allgemeiner Hinweis

Für dieses Übungsblatt benötigt ihr das in StudIP zur Verfügung gestellte Archiv `ueb3-vorgabe.zip` mit Vorgabedateien.

Aufgabe 1 (3 Punkte)

In einem System werden für einen gewissen Zeitraum die eingehenden Unterbrechungen (Interrupts) der verschiedenen Prioritätsstufen (je höher, desto dringender) beobachtet:



a) Wie würden diese Unterbrechungen abgearbeitet, wenn man die unten aufgeführten Bearbeitungszeiten zugrundelegt. Die Interrupts sind zustandsorientiert, gleiche Interrupts können zu jeden Zeitpunkt nur einmal anliegen. Gebt eine Grafik analog zu der Grafik in Folienstapel ti2-05, Folie 46 ab.

Priorität	Bearbeitungszeit
1	1
2	3
3	2
4	1
5	3
6	4

b) Ist der resultierende Ablauf zufriedenstellend (mit Begründung)?

Aufgabe 2 (2 Punkte)

Schreibt auf Basis des Gerüsts in `aufgabe2/sigserver.cc` ein einfaches C++-Programm mit dem Namen `sigserver`, das mittels `sigaction` einen Handler für das Signal `SIGTSTP` installiert. In eurem Signalhandler soll die Nummer des empfangenen Signals ausgegeben werden sowie

die Prozess-ID und die User-ID des *sendenden* Prozesses. Danach wird die Programmausführung fortgesetzt, d. h. der Signalhandler soll das Programm nicht beenden. Das Hauptprogramm besteht aus einer Endlosschleife, in der mittels `pause` auf eingehende Signale gewartet wird.

Hinweis: Benutzt in eurer Lösung die Variante `sa_sigaction` zum Setzen eures Handlers, d. h. `sa_flags` muss auf `SA_SIGINFO` gesetzt werden (siehe `man sigaction`).

Aufgabe 3 (5 Punkte)

Informationen über Prozesse in einem Unix-System können über das spezielle proc-Dateisystem ermittelt werden. Dieses ist üblicherweise unter dem Pfad `/proc` eingebunden.[1] Eintrage zu einem Prozess mit der ID *pid* sind im Unterverzeichnis *pid* von `/proc` zu finden. Wie im Manual zu `proc` beschrieben, liefert die Datei `stat` weitere Informationen zu dem betreffenden Prozess. Jeder Prozess besitzt einen oder mehrere Tasks, die als Unterverzeichnisse unterhalb von `/proc/$pid/task` zu finden sind. Die Datei `/proc/$pid/task/$tid/children` enthält die Prozess-IDs der Kindprozesse des jeweiligen Tasks.

Entwickelt auf Basis des Gerüsts in `aufgabe3/proctree.cc` ein Programm, das die gerade laufenden Prozesse auf einem (Linux-)System ausgibt. Die Kindprozesse eines Prozesses sollen um zwei Leerzeichen nach rechts gegenüber dem Elternprozess eingerückt sein.[2] Die Ausgabe soll die Prozess-ID und den Namen des ausgeführten Kommandos umfassen. Als Test reicht ein einfacher Funktionstest, der eine vergleichbare Ausgabe wie die folgende erzeugen sollte (gekürzt). Geht in Eurer Lösung auch auf mögliche Fehlersituationen bei den Operationen auf dem Dateisystem ein.

```
1 (systemd)
  470 (blkmapd)
  474 (lvmetad)
  550 (rpc.idmapd)
  ...
  1073 (vmware-usbarbit)
  1167 (rpc.mountd)
  1176 (ntpd)
  1180 (sshd)
    2477 (sshd)
      2517 (sshd)
        2519 (bash)
          3105 (proctree)
```

Abgabe

Bis 23:59 Uhr am 16.11.2023 digital in StudIP. **Es gelten die vereinbarten Scheinbedingungen (siehe StudIP).** Bitte beachtet unsere ergänzenden Hinweise ebenda.

Ladet die abgaberelevanten Dateien in DoIT hoch.

Eure Ansätze und der gewählte Lösungsweg müssen nachvollziehbar sein. Achtet insofern auf eine saubere Dokumentation im Quelltext und im abgegebenen PDF-Dokument. Benennt alle von Euch verwendeten Quellen, auch Zusammenarbeit mit anderen Gruppen und verwendete Unterlagen aus früheren Jahrgängen.

[1] Siehe auch `man proc`.
[2] Ähnlich der Ausgabe von `pstree`.

Programmieraufgaben sind im Zweifel in C++20 zu entwickeln. Die Korrektheit der Lösung bzw. deren Grenzen sind grundsätzlich in der Abgabe nachzuweisen. Dies geschieht neben der Dokumentation des Programmcodes in den Quelldateien und zusätzlich in dem abgegebenen PDF-Dokument mit der Lösungsbeschreibung durch geeignete Tests, deren Auswahl und Eignung begründet werden müssen. Als **Referenzplattform** gelten die Linux-Rechner im Rechnerpool in MZH E0.