1. Prioritäten unter Linux: nice

a) In dieser Aufgabe testen Sie die Auswirkung verschiedener Nice-Levels auf Prozesse im Linux-Container. Starten Sie einen Ubuntu-Container mit zusätzlichen Optionen (rot), damit im Container nur eine CPU verfügbar ist und `nice` funktioniert:

```
docker run -it --rm -v ${PWD}:/realworld --cpuset-cpus=0 --privileged=true hgesser/ubuntu-dev
```

Laden Sie den Programm-Quellcode herunter, entpacken Sie das Archiv und wechseln Sie in den Ordner `ue07`:

```
cd /realworld
wget swf.hgesser.de/bs-b1/prakt/bs1-ue07.zip
unzip bs1-ue07.zip
cd ue07
```

Der Ordner enthält die C-Datei `nice-test.c`:

```
[...]  // #include-Befehle
#define _load 500
#define _work { float a=3.1415,b=2.1234; int i,j; for (i=0;i<_load;i++) for (j=0;j<_load;j++) a =
a*i+b*j;  }

int main () {
  int counter;
  char output;
  // struct timeval tv1, tv2;
  // gettimeofday(&tv1, NULL);
  int first = getpid();

  if (fork()) {
    output = 'a';     // 1. Gruppe
  } else {
    if (fork()) {
      output = 'b';   // 2. Gruppe
    } else {
      output = 'c';   // 3. Gruppe
    }
  }
  fork(); fork();     // in jeder Gruppe vier Prozesse

  for (counter = 0; counter < 100; counter++) {
    printf ("%c", output); fflush (NULL);
    _work;
  }

  // gettimeofday(&tv2, NULL);
  // double time_spent = (tv2.tv_usec - tv1.tv_usec) / 1000000.0
  //                   + (tv2.tv_sec  - tv1.tv_sec);
  // printf ("%c:done, time: %f\n", output, time_spent);
  if (first == getpid()) { sleep (10); printf ("\n"); }
}
```

Übersetzen Sie das Programm und starten Sie es mit

```
gcc -o nice-test nice-test.c
./nice-test
```

Wenn Sie ein zweites Terminalfenster öffnen und darin eine Shell im gleichen (!) Docker-Container starten (siehe Hinweis nächste Seite), können Sie sich mit dem Befehl

```
ps -C nice-test
```

davon überzeugen, dass durch die diversen `fork`-Aufrufe insgesamt zwölf Prozesse laufen. (Geben Sie den Befehl ein, während die Prozesse noch aktiv sind.) Jeder dieser Prozesse gibt 100 x „a“, „b“ oder „c“ aus; die Prozesse werden gleich behandelt, darum treten vorne und hinten die Buchstaben gleich häufig auf.

> *Hinweis zu Docker*: In einem zweiten Terminalfenster können Sie auf den laufenden Docker-Container mit Debian Linux zugreifen, indem Sie `docker ps` eingeben. Falls Sie mehrere Container nutzen, erscheinen mehrere Zeilen; der gesuchte Container hat `hgesser/ubuntu-dev` in der Spalte `IMAGE`. Kopieren Sie dann die `CONTAINER ID`, die in der ersten Spalte steht (z. B. `2eef9e276ca2`) und starten Sie dann eine neue Shell, die in diesem Container läuft, mit
>
> ```
> docker exec -it 2eef9e276ca2 bash
> ```

b) Bauen Sie in die geschachtelte Fallunterscheidung für die drei Gruppen `nice()`-Aufufe auf, so dass

- die Prozesse der ersten Gruppe („a“) mit erhöhter Priorität,
- die Prozesse der zweiten Gruppe („b“) mit normaler Priorität und
- die Prozesse der dritten Gruppe („c“) mit reduzierter Priorität

laufen. Eine Beschreibung der `nice()`-Funktion erhalten Sie über

```
man 2 nice
```

Wählen Sie dabei jeweils Extremwerte. Die größten bzw. kleinsten zulässigen Werte verrät die Manpage nicht, aber Sie finden die Antwort in der Manpage des gleichnamigen Shell-Befehls:

```
man 1 nice            # Beachte: 1 statt 2 ...
```

Übrigens: Es handelt sich bei `nice()` um keinen (!) System Call, `nice()` ruft `setpriority()` auf, und das ist ein System Call, siehe Definition von `__NR_setpriority` in `unistd_64.h`.

Übersetzen und starten Sie das Programm erneut und beobachten Sie das geänderte Verhalten – die „a“s sollten jetzt weiter vorne konzentriert, die „c“s stärker am Ende sein.

c) Prüfen Sie, ob das Setzen der Nice-Levels in allen Prozessen erfolgreich war. Das geht mit

```
ps -o pid,ni,cmd -C nice-test
```

Über die Option `-o pid,ni,cmd` zeigt `ps` in drei Spalten die Prozess-ID (`pid`), den Nice-Level (`ni`) und den Prozessnamen (`cmd`; Kommando) an.

Starten Sie das Programm erneut, diesmal aber mit `sudo` und damit mit Root-Rechten:

```
sudo ./nice-test
```

und untersuchen Sie erneut die Nice-Levels. Können Sie den Unterschied zum Aufruf ohne Root-Rechte erklären?

> *Hinweis zu Docker*: Docker erlaubt negative Nice-Werte für Prozesse nur, weil Sie den Container mit der zusätzlichen Option `--privileged=true` gestartet haben, s.o.
>
> Probieren Sie es auch einmal ohne diese Option; dazu müssen Sie den Container beenden und neu starten. Welche Änderung sehen Sie?

d) Die Beobachtung der Ausgaben („a“, „b“, „c“) gibt nur einen groben Eindruck davon, welche Auswirkungen die Nice-Levels auf die Priorisierung haben. Für ein quantitatives Ergebnis kommentieren Sie die Zeile

```
printf ("%c", output); fflush (NULL);
```

aus und entfernen die Kommentarzeichen `//` vor den sechs bisher auskommentierten Zeilen. Die so veränderte Version gibt für jeden Prozess Laufzeit-Messwerte aus. Schätzen Sie den Faktor, um den Prozesse mit höchster Priorität schneller laufen als solche mit geringster Priorität.

Wenn Sie erfahren möchten, wie die Laufzeitberechnung funktioniert, lesen Sie die Manpage der zweimal aufgerufenen Funktion `gettimeofday()`. Hinweis dazu: Eine Mikrosekunde (1 µs) = 10^{-6} Sekunden (10^{-6} s) bzw. 1 s = 10^{6} µs 1.000.000 µs.