6. Prozessliste: ein ps-Klon

a) In dieser Aufgabe geht es darum, eine Prozessliste auszugeben. Dabei greifen Sie auf die Informationen zurück, die Sie aus dem `/proc`-Dateisystem (aus den Einträgen `/proc/PID/stat`) auslesen können. Ihre Version von `ps` soll tabellarisch die folgenden Informationen für alle Prozesse ausgeben:

- Prozess-ID
- Status (ein Buchstabe)
- Parent-Process-ID
- Prozessgruppen- und Session-IDs
- Terminal (nur numerisch, es ist keine Umwandlung in `tty1`, `tty2`, `pts/1` etc. nötig)
- Kommando in Langfassung – das erhalten Sie nicht aus `/proc/PID/stat`, sondern über die Datei `/proc/PID/cmdline`.

Das Hauptprogramm besteht aus einer Schleife, die von 1 bis 32768 läuft und für jede dieser potenziellen Prozess-IDs (diese Grenze finden Sie über

```
# cat /proc/sys/kernel/pid_max
32768
```

heraus) prüft, ob es ein Verzeichnis `/proc/pid/` gibt:

1. Öffnen Sie das Verzeichnis `/proc/pid` mit `open()`.
2. Im Erfolgsfall schließen Sie den File Descriptor direkt wieder und wissen, dass es dieses Verzeichnis gibt.
3. Dann öffnen Sie die Datei `/proc/pid/stat` und kopieren den Inhalt dieser Datei in einen Puffer. (500 Byte Puffergröße reichen aus.)
4. Der Pufferinhalt besteht aus mehreren Feldern, die durch Leerzeichen getrennt sind. Sie können sich eine kleine Routine `getvalue()` schreiben, die das `i`-te Feld zurück gibt (also `i` Leerzeichen überspringt).
5. Mit den folgenden `#define`-Anweisungen

   ```
   #define PROC_PID 0
   #define PROC_NAME 1
   #define PROC_STATE 2
   #define PROC_PPID 3
   #define PROC_PGRP 4
   #define PROC_SESS 5
   #define PROC_TTYNO 6
   #define PROC_UTIME 13
   #define PROC_STIME 14
   #define PROC_NICE 18
   #define PROC_THREADS 19
   ```

   finden Sie schnell die richtigen Felder und übertragen die Inhalte in dafür vorbereitete String-Variablen, die Sie schließlich mit

   ```
   printf ("%7s  %-2s%7s  %7s  %7s  %4x %s\n",
           prid, state, ppid, pgrp, sess, ttyno, longcmd);
   ```

   ausgeben. Fügen Sie vorab noch eine Überschriftszeile ein.

Über die Vorlesungswebseite erreichen Sie eine Quellcode-Datei (`myps.c`), die schon die obigen Definitionen enthält und alle benötigten Header-Dateien einbindet.

Um aus einer Integer-Zahl `pid` den String `"/proc/pid/"` bzw. `"/proc/pid/stat"` zu erzeugen, benötigen Sie die Funktion `sprintf()`, die ähnlich arbeitet wie `printf` (siehe `man sprintf`).

Nach ersten Experimenten werden Sie feststellen, dass `/proc/PID/cmdline` bei einigen Prozessen leer ist. Diesen Fall sollen Sie erkennen und dann stattdessen das `cmd`-Feld aus `/proc/PID/stat` ausgeben; dann geben Sie den Kommandonamen in eckigen Klammern (z. B. `[kthreadd]`) aus, wie es auch das Kommando `ps` tut.

b) Starten Sie Firefox und öffnen Sie ein paar Webseiten; das Programm besteht aus mehreren Threads. Was fällt Ihnen (anhand von Firefox) auf, wenn Sie die Prozessliste mit Ihrer ps-Implementation anzeigen?