Aufgabe 7: Signale

Mit Hilfe von Signalen ist es dem Betriebssystem möglich, einem Prozess einfache Informationen mitzuteilen. Da die meisten Signale versendet werden, wenn etwas schief gegangen ist, wird das Programm in der Standardkonfiguration beendet. Dieses Verhalten lässt sich jedoch für fast alle Signale überschreiben. So kann z.B. ein Programm auf Fehler reagieren oder Debug-Informationen ausgeben.

In den folgenden Teilaufgaben sollen Sie die Informationen, die beim Absturz Ihres Programms ausgegeben werden, verbessern. Im Speziellen soll beim Erhalt eines Signals eine Liste mit Namen aller aufgerufenen, aber noch nicht beendeten Funktionen ausgegeben werden. Sie kennen diese Funktionalität vermutlich bereits vom Debugger-Befehl *backtrace* oder vom *address-sanitizer*.

Als Vorbereitung sollten Sie sich folgende Man-Pages durchlesen: **signal(7)**, **signal(3)** sowie **backtrace(3)**.

Beachten Sie, dass einige Bibliotheksfunktionen `errno` setzen. Implementieren Sie in diesem Fall eine Fehlerbehandlung.

7.1 Backtrace: Signal Handler (2P)

Implementieren Sie die Funktionen `initialize_signal_handler()` sowie `signal_handler()` in der Datei `Backtrace/src/backtrace.c`. Die Funktion `initialize_signal_handler()` soll *nur* für die nachfolgenden Situationen die Funktion `signal_handler()` als Verarbeitungsroutine registrieren.

- Interrupt durch Tastatur
- Beenden durch Tastatur
- Ungültige Maschinenanweisung
- Terminierung
- Abnormale Prozessterminierung
- Floating Point Exception
- Segmentation Fault

Beachten Sie, dass Sie *exakt* diese sieben Signale abfangen. Geben Sie dann in `signal_handler()` die Signalnummer aus und beenden Sie den Prozess mit der Signalnummer als Exitstatus.

Testen: Ob Ihre Implementierung funktioniert, sehen Sie, wenn Sie das Testprogramm `backtrace` z.B. mit `STRG + c` oder dem Programm **pkill(1)** terminieren. Weiterhin steht Ihnen auch das Skript `test-backtrace.sh` zur Verfügung, das auch in der GitLab CI unter `run-test-backtrace` ausgeführt wird.

7.2 Backtrace: Funktionsaufrufe ausgeben (4P)

Implementieren Sie mit Hilfe von **backtrace(3)** die Funktion `print_backtrace()` und rufen Sie diese aus `signal_handler()` auf. Wie der Name schon vermuten lässt, soll `print_backtrace()` eine Liste der aktiven Funktionen ausgeben. Verwenden Sie das in `backtrace/include/backtrace.h` definierte Makro für den Aufruf von **backtrace(3)**. Die Ausgabe Ihres Programms soll sich an der nachfolgenden Ausgabe orientieren. Das heißt, ihre Ausgabe soll zumindest *ähnlich* aussehen und selbsterklärend sein.

```
Received Signal: 2
-------------------------------------------------------
[DEPTH] FILE(SYMBOL+OFFSET) [ADDRESS]
-------------------------------------------------------
[00] ./test() [0x400b09]
[01] /lib/x86\_64-linux-gnu/libc.so.6(_libc_start_main+0xf5) [0x7f2e755e5ec5]
[02] ./test(main+0x68) [0x400df8]
[03] ./test(decision_maker+0x7b) [0x400d4b]
[04] ./test(tick_two+0x3b) [0x400c6b]
[05] /lib/x86_64-linux-gnu/libc.so.6(sleep+0xd4) [0x7f2e75685724]
[06] /lib/x86_64-linux-gnu/libc.so.6(nanosleep+0x10) [0x7f2e75685870]
[07] /lib/x86_64-linux-gnu/libc.so.6(+0x36da0) [0x7f2e755fada0]
[08] ./test(signal_handler+0x2f) [0x400f6f]
[09] ./test(print_backtrace+0x1c) [0x400e1c]
```

Testen: Auch bei dieser Aufgabe können Sie die Implementierung testen, indem Sie das Programm `backtrace` mit `STRG + c` beenden oder das Skript `test-backtrace.sh` verwenden. In der GitLab CI steht Ihnen auch hier der Test `run-test-backtrace` zur Verfügung.

7.3 Send-signal: Signale an andere Prozesse schicken (4P)

In Aufgabe 7.1 haben Sie für verschiedene Signale ein eigenes Verhalten implementiert. Jetzt sollen Sie in `Send-signal/src/main.c` ein Programm schreiben, dass Signale mit **kill(3)** an beliebige Prozesse verschickt.

Das Programm `send-signal` soll mit *exakt* zwei Argumenten von der Kommandozeile aus aufgerufen werden. Um die Bedienung von `send-signal` zu vereinfachen, sollen alle Argumente mit Hilfe von *Optionselementen* übergeben werden, sodass sie in beliebiger Reihenfolge übergeben werden können. Optionselemente beginnen mit einem `-` gefolgt von einem einzelnen Buchstaben. Je nach Option kann danach noch ein beliebiges Argument folgen, das vom Programm gelesen wird. Ein Beispiel, das Sie alle kennen, ist `git commit -m "ein Kommentar"`. Implementieren Sie die Optionselemente mit **getopt(3)** und das Versenden von Signalen mit **kill(3)**.

Das Programm `send-signal` soll **exakt** zwei Argumente bekommen:

- `-s SIGNALNUMMER`
- `-p PROZESS_ID`

Beispiel: `send-signal -p 5982 -s 9`

Hält sich der Benutzer nicht an die vorgegebene Bedienung, soll über die Funktion `print_usage()` ein Hinweis zur korrekten Nutzung des Programms ausgegeben werden.

Hinweise: Beachten Sie, dass Argumente als String übergeben werden. Eine Umwandlung in Integer können Sie mit **atoi(3)** erreichen. Die Prozess-IDs Ihrer Prozesse bekommen Sie mit dem Aufruf `ps -u`.

Testen: Testen Sie Ihre Implementierung, indem Sie verschiedene Signale mit `send-signal` an `backtrace` senden, und nutzen Sie das Testprogramm `test-signal.sh`. Der entsprechende Test in der GitLab CI ist unter `run-test-signal` verfügbar.

Abgabe bis 21.12.2018, 23:59 Uhr.

Ihre Lösungen sollen bis zur Deadline im GitLab des IBR hochgeladen sein.
Einhaltung der Coding Guidelines: 2 Punkte
Zu erzielende Minimalpunktzahl: 6 Punkte