Technische Universität Berlin Fachgebiet Distributed and Operating Systems Sommersemester 2024	**Praxisaufgabe 1** zu – Systemprogrammierung – Prof. Dr. Odej Kao
Abgabetermin:	**14.06.2024 23:59 Uhr**

In dieser Aufgabe sollen verschiedene Schedulingverfahren implementiert werden. Ein einfaches FCFS-Beispiel ist bereits vorgegeben. Beachten Sie auch die README innerhalb der Vorgabe!

Aufgabe 1: Queue (2P)

Erstellen Sie eine Queue. In den Dateien queue.c und queue.h sind dafür bereits die nötigen Funktionen und Structs vorgegeben. Beachten Sie die Dokumentation in der Headerdatei. Für einige der Schedulingverfahren kann diese Queue hilfreich sein, für andere Verfahren sind andere Datenstrukturen vermutlich nützlich.

Aufgabe 2: Scheduler (8P)

Implementieren Sie anschließend die verschiedenen Scheduler, wie in den Vorgaben spezifiziert. Es sollen die Verfahren LCFS und HRRN als Beispiele für nicht verdrängende sowie PRIO-P, SRTN, RR und MLF (in abgeänderter Form – siehe Hinweis) als Beispiele für verdrängende Algorithmen implementiert werden. Studieren Sie dazu auch die scheduler.c und scheduler.h Dateien um sich einen Überblick zu verschaffen.

Hinweis zum MLF-Verfahren: Abweichend zum klassischen MLF-Verfahren aus den Folien sollen Sie hier eine kleine Besonderheit in das Scheduling-Verfahren einbauen: Innerhalb der Level-Warteschlangen soll, anstatt FCFS zu verwenden, nach Priorität geordnet werden. Beim klassischen MLF werden die Prozesse innerhalb der Level danach sortiert, zu welchem Zeitpunkt sie in dieses Level gekommen sind, sprich innerhalb der Level-Warteschlangen gilt FCFS bzw. FIFO. Für diese Aufgabe sollen Sie hier aber die Prioritäten der Prozesse anstatt der Ankunftszeit im Level als entscheidenden Faktor nutzen, welcher Prozess von mehreren aus dem gleichen Level als nächstes ausgewählt wird.

Beispiel: Die Prozesse A, B und C sind in dieser Reihenfolge ins Level 1 gekommen. Außerdem gelten folgende Prioritäten $(A = 2, B = 6, C = 4)$. Ignorieren wir einmal alle anderen Prozesse. Normalerweise würde erst A, dann B und dann C sein Quantum erhalten, aber für unsere Aufgabe würde in einem solchen Fall erwartet werden, dass hier B zuerst rankommt, dann C und zum Schluss A.

Diese Form des MLF-Verfahrens soll mit insgesamt 4 Leveln implementiert werden, wobei die jeweilige Zeitscheibe $\tau = 2^i$ beträgt. Das erste Level hat den Index $i = 0$, das nächste $i = 1$ und so weiter. Das letzte Level soll nach PRIO-NP arbeiten, das heißt, dass Prozesse, die dieses Level erreichen und dann (nach Priorität) gescheduled werden, ihre gesamte Restlaufzeit abarbeiten können.

Es kann hilfreich sein, weitere Hilfsfunktionen zu erstellen.

optional: Erstellen Sie weitere Testcases. Auch hier können Sie sich an den Vorgaben orientieren.

Hinweise:

- **Beschreibungen der Funktionen:** Nähere Beschreibungen der einzelnen Funktionen finden Sie auch in den .h-Dateien im Ordner include.
- **Vorgaben:** Bitte ändern Sie bestehende Datenstrukturen, Funktionsnamen, Headerfiles, ... nicht. Eine Missachtung kann zu Punktabzug führen. Gerne können Sie weitere Hilfsfunktionen oder Datenstrukturen definieren, die Ihnen die Implementierung leichter gestalten.
- **Makefile:** Bitte verwenden Sie für diese Aufgabe das Makefile aus der Vorgabe. Führen Sie hierzu im Hauptverzeichnis das *make* aus. *make* kompiliert das Projekt mit *clang*. Unter Ubuntu können *make* und *clang* mit dem Befehl *sudo apt install make clang* installiert werden. Durch den Befehl *make clean* werden kompilierte Dateien gelöscht. Gerne können Sie das Makefile um weitere Flags erweitern oder anderweitig anpassen. Ihr Programm sollte aber mit dem vorgegebenen Makefile weiterhin compilierbar bleiben. Auch hier sei erneut auf die README hingewiesen.
- **Memoryleaks:** Überprüfen Sie abschließend Ihr Programm auf Memory leaks, um zu evaluieren ob der gesamte vom Scheduler allokierte Speicher wieder freigegeben wurde. Hier empfiehlt sich das Kommandozeilenwerkzeug valgrind[1]. Memory leaks führen zu Punktabzug.

Abgabe:

- **Verpacken** sie den Ordner *src* mit den bearbeiteten **.c* Dateien (wie in der Vorgabe) zu einem zip-Archiv mit dem Namen *submission.zip*. Header-files (**.h*) sollen nicht verändert und damit auch nicht mit abgegeben werden. Hierfür können sie das make target *make pack* verwenden. Damit *make pack* funktioniert muss das tool *zip* auf ihrem System installiert sein! Das Archiv laden sie dann in ISIS hoch.
- **Wichtig:** Es ist nicht notwendig, das Archiv und dessen Inhalt mit ihrem Namen bzw. Matrikelnummer zu personalisieren. Die Abgabe wird automatisch ihrem ISIS Account zugeordnet!

[1] http://valgrind.org/