LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Grundlagenpraktikum: Rechnerarchitektur

Arbeitsblatt 7
06.06.2022 - 12.06.2022

T7.1 File IO

In dieser Aufgabe wollen wir uns dem Öffnen und Lesen von Dateien widmen. Hierzu werden wir das Programm `toupper_simd` so erweitern, dass der Nutzer auch eigene Eingabedateien spezifizieren kann.

Hinweis: Ziehen Sie ggf. die relevanten `man` Pages für weitere Informationen heran. Achten Sie auch stets auf eine geeignete Fehlerbehandlung und hilfreiche Fehlermeldungen.

Vorlage: `https://gra.caps.in.tum.de/m/toupper_simd.tar` – Für die Bearbeitung der Aufgabe müssen Sie lediglich die Funktionen `read_file` und `write_file` bearbeiten.

1. Implementieren Sie zunächst die Funktion `read_file`. Öffnen Sie hierzu mit `fopen` die Datei, deren Pfad der Funktion übergeben wurde.
2. Finden Sie heraus, wie Sie die Funktion `fstat` verwenden können, um die Größe der Datei zu bestimmen (`man 2 fstat`). Wie können Sie zudem feststellen, ob es sich bei der Datei um eine reguläre Datei handelt?
3. Allozieren Sie nun genügend Speicher für den Dateiinhalt und lesen Sie die Datei mit `fread` ein. Was müssen Sie hierbei mit Hinblick auf das terminierende `NULL`-Byte beachten?
4. Bevor sie den String zurückgeben, denken Sie daran, die geöffnete Datei wieder mit `fclose` zu schließen. Achten Sie darauf, dass die Datei auch im Fehlerfall geschlossen wird!
5. Implementieren Sie nun die Funktion `write_file`. Öffnen Sie hierzu auch in dieser zunächst die Ausgabedatei.
6. Schreiben Sie nun den String *ohne* das terminierende `NULL`-Byte in die Datei. Denken Sie auch hier daran, die Datei in jedem Fall zu schließen.

S7.1 Zeitmessung

Um verschiedene Implementierungen zur Lösung eines Problems vergleichen zu können, wird oftmals die benötigte Laufzeit als Kriterium herangezogen. Bei diesem vermeintlich einfachen Problem gibt es jedoch einige Fallstricke, welche beim Benchmarking beachtet werden sollten[1].

[1] Das trifft selbstverständlich auch auf Ihre Projektaufgabe zu.

- Die C-Standardbibliothek (bzw. Erweiterungen aus dem POSIX-Standard und Linux-spezifische Funktionen) enthält verschiedene Funktionen, um die Zeit zu messen. Wie unterscheiden sich die Funktionen `clock_gettime`, `timespec_get` (Achtung: hat noch keine `man`-Page), `gettimeofday` und `clock`? Welche von diesen Möglichkeiten ist empfehlenswert?
- Warum ist die Verwendung des Befehls `time` oftmals nicht empfehlenswert?
- Weshalb ist das Messen von I/O-Operationen wie `printf` problematisch? Lesen Sie hierzu auch `man 7 pipe` und bedenken Sie, dass auch `stdout` nicht notwendigerweise ein Terminal (auch eine *pipe*) ist, sondern auch ein anderes Programm, ein Socket oder eine Datei sein kann.
- Überlegen Sie, aus welchen Gründen es zu Ungenauigkeiten bei der Zeitmessung kommen kann – beispielsweise, wenn die Laufzeiten mehrerer Durchläufe stark schwanken. Wie kann man diese Probleme vermeiden?

P7.1 SIMD-Anwendung: ToUpper [3 Pkt.]

In dieser Einheit betrachten wir die Funktion `toupper`, welche in einem String sämtliche Klein-Buchstaben durch die entsprechenden Groß-Buchstaben ersetzt. Optimieren Sie diese Funktion mit den *SSE*-Erweiterungen.

P7.2 SIMD-Anwendung: strlen [3 Pkt.]

SIMD kann, wie Sie bereits in den Videos gesehen haben, mit General Purpose Registern durchgeführt werden. Berechnen Sie mit Hilfe von SIMD auf General Purpose Registern die Länge des Strings `str`.

```
size_t strlen(const char* str);
```

Sie haben für diese Aufgabe keinen Zugriff auf SSE. Außerdem ist die Anzahl an verfügbaren Zyklen begrenzt.

Q7.1 Quiz [4 Pkt.] (siehe Praktikumswebsite)