Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Betriebssysteme Praktikum 3

Protokoll

Florian Heuer

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Aufgabenstellung

1.1 Beschreibung

Drei verschiedene Seitenersetzungsalgorithmen sollen in diesem Laborversuch implementiert werden. Diese sollen FIFO-, LRU- und der CLOCK- Algorithmus sein. Um diese zunächst implementieren zu können sollen 2 C Programme erstellt werden die auf einen gemeinsamen Speicher zugreifen. Der gemeinsame Speicher (Shared Memory) soll dabei den Hauptspeicher eines realen Systems simulieren. Zusätzlich hält der Shared Memory einige Administrationsdaten auf die beide Programme zugreifen müssen.
Das erste Programm erstellt zufällige Zahlen, schreibt diese in den „Hauptspeicher“ (Shared Memory), liest diese wieder aus und sortiert diese im Anschluss. Das schreiben und lesen soll hier implementiert werden. Erstellung, sortieren und der Ausgabe sind bereits vorgegeben.
Damit der „Hauptspeicher“ nicht überschrieben wird, wenn dieser zu knapp wird, soll das 2. Programm dieses managen. D. h. nicht benötigte Frames des „Hauptspeichers“ werden in eine Auslagerungsdatei geschrieben und bei Bedarf wieder ausgelesen. Wie diese Frames ausgewählt werden, ist Hauptbestandteil des Programms. Hier sollen die oben genannten Algorithmen implementiert werden. Die Kommunikation beider Programme erfolgt mittels asynchroner Signale und Handlern, welche auf diese reagieren. Wie dieses umgesetzt werden soll ist im Detail in 2.2 beschrieben.

2 Entwurf

2.1 Umgebung

Das Programm wird in Sprache C entwickelt. Als Compiler kommt „GCC“ zum Einsatz. Das Betriebssystem Open SUSE in Version 12.3 wird zum kompilieren und ausführen des Codes benutzt. Als Editor wird Sublime 2 verwendet. Zudem wird zur Erleichterung des Kompiliervorgangs ein Makefile erstellt.

2.2 Umsetzung

Wie in Punkt 1.1 erwähnt werden für die Umsetzung 2 C Programme benötigt. Eines welches die Hauptapplikation bildet, genannt **vmapp** und eines das den „Hauptspeicher“ verwaltet, genannt **mmange.** Im folgenden wird das Implementierungskonzept beider Programme beschrieben.

2.2.1 Programm 1 (vmapp)

Dieses Programm besteht aus Sourcedateien. Eines stellt die eigentliche Applikation dar, welche eine Reihe von Zufallszahlen definiert, diese in den Hauptspeicher schreibt, ausließt und sortiert. Dieses war vorgegeben und bedarf keiner Implemtierung.

Die 2. Sourcedatei implementiert 3 Methoden. Jeweils eine zum lesenden und schreibenden Zugriff auf den Shared Memory und eine Methode zum Verbinden mit dem Shared Memory, das erzeugen des Shared Memory erfolgt in Programm 2. Die Methoden zum Lesen und Schreiben greifen auf den gemeinsamen Speicher zu, indem Sie aus der übergebenen virtuellen Adresse eine Page und einen Offset berechnen, danach wird geprüft ob dieser im „Hauptspeicher“ liegt. Wenn nicht wird ein Signal (kill Aufruf mit Prozess ID von Programm 2 und Parameter SIGUSR1) an Programm 2 gesendet. Dieses sorgt dann dafür, dass die angefragte Page in den Speicher geladen wird. Damit das Programm genügend Zeit dafür besitzt, blockiert hier der gesamte Prozess (Blockieren auf einer gemeinsamen Semaphore, die über den Administrationsteil des Shared Memory geteilt wird) und wird erst fortgesetzt, bis Programm 2 die Semaphore freigibt. Anschließend wird der Frame ermittelt, in dem Page nun liegt. Mit dem Frame und dem zuvor berechneten Offset ist es nun möglich die Adresse im „Hauptspeicher“ zu berechnen und den Wert auszulesen bzw. hinein zu schreiben.

2.2.2 Programm 2 (mmanage)

Kurzum hört das Programm auf Signale, reagiert entsprechend, erstellt und initialisiert den Shared Memory(„Haupspeicher“). Des Weiteren erzeugt es eine binäre Auslagerungsdatei und ein Logfile. Auf folgende Signale reagiert das Programm: SIGUSR1, SIGUSR2, SIGINT. Die Reaktion erfolgt mittels Handler. Im Fall von **SIGUSR1** (Pagefault) kann Programm 1 nicht auf die angeforderte Page zugreifen. Hier wird nun ein Frame mittels Algorithmus(Clock, LRU, FIFO) bestimmt, in welches die angefragte Page geladen werden soll. Die zuvor gespeicherte Page in dem Frame wird insofern sie verändert wurde ,(DIRTY FLAG) ausgelagert und in ein externes binäres File geschrieben. Die angeforderte Seite kann nun aus dem binärem File (Auslagerungsdatei) geholt und in das Frame geschrieben werden. Nun wird die Pagetable verändert und das PRESET_FLAG gesetzt. Die in Programm 1 geblockte Semaphore wird frei gegeben. Programm 1 kann nun weiter arbeiten. Im Fall **SIGUSR2** wird ein Dump des virtuellen Speichers ausgegeben. Bei dem Signal **SIGINT** wird das Programm beendet und der Shared Memory gelöscht.

3 Abnahme

Der folgende Abschnitt soll alle vom Prüfer ausgehenden Anmerkungen bzw. geforderte Anpassungen am präsentierten Programm protokollieren.

3.1 Allgemein

Die Abnahme des Programms wurde erfolgreich am 29.11.2016 von Prof. Dr. Fohl im Labor 0761 durchgeführt. Im Zuge der Abnahme sind einige Änderungen am Code resultiert, welche im folgenden Punkt aufgeführt werden.

3.2 Anmerkungen & Anpassungen

Die unterhalb aufgeführten Punkte beschreiben die getätigten Änderungen am Quellcode.

3.2.1 Überarbeitung der Algorithmen

Wie während der Abnahme zur Unzufriedenheit festgestellt wurde, wichen die ermittelten Pagefaults etwas von den Ergebnissen der Musterlösung ab. Um ca. 15 Pagefaults wichen die Ergebnisse für alle 3 Algorithmen ab. Des Weiteren haben sich die Algorithmen nicht ausschließlich um die Bestimmung des nächst zu allokierenden Frames gekümmert, sondern zusätzlich das Auslagern der Seiten und Aktualisieren der Pagetable vorgenommen. Dadurch ist redundanter Code resultiert.

Zur Lösung der oben beschriebenen Mängel wurde der Code überarbeitet. Die Ergebnisse sind dem Repository zu entnehmen und beschränken sich ausschließlich die Datei mmanage.c. Die Algorithmen sind nun über die globale Variable ***„vmem_algo“*** einstellbar.

Die Werte der Pagefaults nähern sich nun stark den Vorgaben. Im folgendem sind die ermittelten Ergebnisse aufgeführt.

	Pagefaults ermittelt	**Pagefaults erwartet** (Musterlösung)
LRU	531	530
CLOCK	540	536
FIFO	556	557

Tabelle 1: Ermittelte Pagefaults

3.2.2 Fehlermeldungen

Des Weiteren wurden einige Debugmeldungen nicht wie gewünscht über das Makro ausgegeben. Eine Fehlermeldung falls mmange nicht gestartet wurde ist nicht erschienen, wenn man vmapp aufgerufen hat.
Debugmeldungen werden nun durch das vorgegebene Makro ausgegeben. Ein Hinweiß, falls der Shared Memory nicht verfügbar ist wird nun korrekt ausgegeben.

4 Fazit

Die Aufgabe 3 des Praktikums hat mir persönlich gut gefallen, da es das Verständnis von Seitenersetzungsstrategien deutlich vertieft und mir den Umgang mit Signalen und Shared Memory aufgezeigt hat.

5 Repository

Der komplette Sourcecode des Programms "**virtualMemoryManagement**" ist unter folgendem Link auf GitHub zu finden.

https://github.com/FlowwX/virtualMemoryManagement

6 Anhang

1. Quellcode (Zip-Archiv), Alternativ: oben genanntes Repository