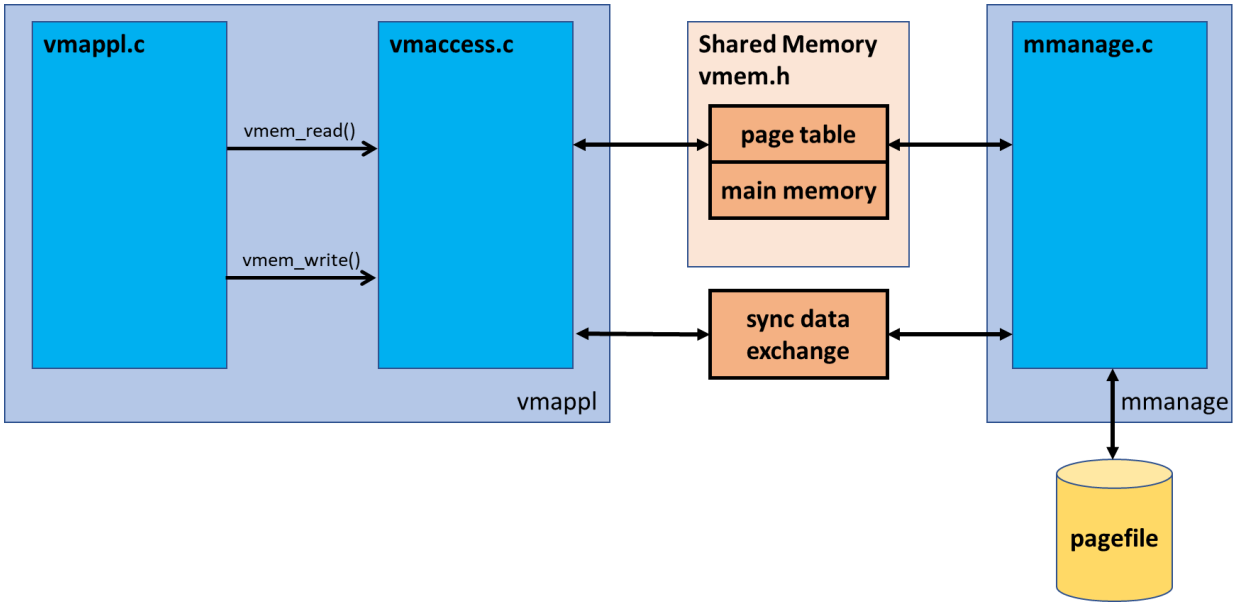
Virtuelle Speicherverwaltung

Name : _____	**Name :** _____
Vorname : _____	**Vorname :** _____
Matrikel-Nr : _____	**Matrikel-Nr :** _____

1. Aufgabenbeschreibung

Diese Aufgabe untersucht Mechanismen der virtuellen Speicherverwaltung. Es wird eine Simulation für die Seitenersetzungsalgorithmen **FIFO**, **CLOCK** und **Aging** erstellt. An die Stelle des physikalischen Speichers eines realen Systems tritt hier ein Speicherbereich im Shared Memory. Über einen weiteren gemeinsamen Speicher sendet die Anwendung Aufträge an den Memory Manager (`syncdataexchange.c`). Über zwei Semaphore wird der Datenaustausch als synchrone Kommunikation realisiert. Neben der Lösung der Aufgabe selbst gibt es die realitätsnahe Anforderung, dass Sie existierenden Code anpassen müssen.

Bitte arbeiten Sie folgende kleine Einführung durch. Lesen Sie anschließend die Dokumentation des Codes und gleichen Sie diese mit der Einführung ab. Das wird Ihnen viel Zeit ersparen.



1. Die Abbildung stellt die beiden Prozesse `vmappl` und `mmanage` dar.
   a. `vmappl` ist eine Anwendung, die ein Feld von ganzen Zahlen wahlweise mit QuickSort oder BubbleSort sortiert.
   b. `mmanage` implementiert die virtuelle Speicherverwaltung.

2. Die Prozesse kommunizieren über zwei Shared Memory Bereiche. Der `SHMKEY` des ersten Bereichs ist `./src/vmem.h`. Die zugehörige C Datei lautet `vmem.h`. In diesem Bereich liegt zum einen das `mainMemory`. Es repräsentiert den Hauptspeicher und ist `VMEM_PHYSMEMSIZE` Byte groß. Der Hauptspeicher ist in Seitenrahmen der Größe `VMEM_PAGESIZE` unterteilt. In der Realität liegt der gesamte virtuelle Speicher auf der Festplatte. Dies wird durch die Datei `pagefile` nachgebildet. Die Größe des virtuellen Speichers beträgt `VMEM_VIRTMEMSIZE` Byte. Er ist in Seiten der Größe `VMEM_PAGESIZE` unterteilt. Diese werden in der Datei `pagefile` hintereinander abgelegt. Somit nimmt der Prozess `mmanage` eine Seite aus der Datei `pagefile` und lagert sie in einen Seitenrahmen von `mainMemory` ein. Wenn eine Seite aus einem Seitenrahmen verdrängt wird, dann schreibt `mmanage` die modifizierte Seite an die richtige Stelle in der Datei `pagefile` zurück.

3. Weiterhin liegt die Seitentabelle (page table) im Shared Memory mit dem `SHMKEY ./src/vmem.h` . Die Seitentabelle wird vom Prozess `mmanage` gepflegt und von dem Modul `vmaccess` , das im Prozess `vmappl` eingebunden ist, gelesen.

4. Der Prozess `vmappl` stellt eine Anwendung dar. Sie füllt zuerst ein Feld mit Zufallszahlen und sortiert es anschließend. Das Feld liegt im virtuellen Speicher. Somit lädt `mmanage` die jeweils benötigten Seiten in einen Seitenrahmen. Der zweite Teil steht im Modul `vmaccess.c` und bildet über die beiden Funktionen `vmem_read` und `vmem_write` die Schnittstelle zum Speicher.

5. Das zweite Shared Memory wird im Modul `syncdataexchange.c` implementiert. Über diesen Mechanismus werden Aufträge an den Memory Manager geschickt. Über zwei Semaphore wird die Kommunikation zwischen Anwendung und Memory Manager synchronisiert.

6. Weiterhin müssen `vmem_read` und `vmem_write`, die die Umrechnung von virtuellen - in physikalische Adressen vornehmen und den `mmanage` über einen PageFault informieren. Ist dies der Fall, blockt `vmem_read` bzw.

vmem_write bis mmanage mitteilt, dass die zum PageFault gehörige Seite in einem Seitenrahmen eingelagert ist und die Seitentabelle aktualisiert wurde.

2. C-Implementierung

Allgemein soll das System aus folgenden Dateien bestehen

1. Das Modul vmappl.c stellt das Anwendungsprogramm dar.
   a. Es werden zufällige Daten erzeugt, angezeigt, mit Quicksort oder Bubblesort sortiert und erneut angezeigt.
   b. Der Parameter seed initialisiert den Zufallszahlengenerator

2. Das Modul vmaccess.c bildet die Schnittstelle zum virtuellen Speicher. Im realen System ist dies die Adress-Decodierungseinheit (Mapping von virtuellen auf reale Adressen).
   a. Die Methoden vmem_read und vmem_write berechnen aus der virtuellen Speicheradresse die Frame-Nummer und den Offset.
   b. Ist die benötigte Seite nicht geladen, wird die Speicherverwaltung (mmanage.c) über einen Auftrag (syncdataexchange.c) aufgefordert die Seite aus dem pagefile in den Hauptspeicher zu laden. Sie sucht einen freien Seitenrahmen, lagert ggf. eine Seite entsprechend den Algorithmen FIFO, CLOCK oder Aging aus und liest die gewünschte Seite ein.
   c. Die Routinen aus vmaccess.c blockieren so lange bis mmanage die synchrone Kommunikation mit einem ACK abgeschlossen hat.

3. Das Modul mmanage.c ist für die Verwaltung der im Hauptspeicher vorhandenen Seitenrahmen zuständig.
   a. Nach der Initialisierung wartet das Modul auf Aufträge von vmappl.
   b. Den Zugriff auf die pagefile realisiert dabei das Modul pagefile.c; ein entsprechendes File ist in TEAMs hinterlegt.
   c. Beim Start initialisiert mmanage.c das Shared Memory, die synchrone Kommunikation, installiert mit sigaction() die Signalhandler, erzeugt bei Bedarf die Datei pagefile und initialisiert die Datenstrukturen. Ein Teil dieser Funktionen finden Sie im Modul mmanage.c fertig implementiert.
   d. Außerdem ruft mmanage.c bei jedem Seitenfehler die Methode logger() auf, die jede Aktion im Logfile protokolliert. **Diese Methode sollte nicht geändert werden.** So kann man Logfiles mit diff vergleichen; ein entsprechendes Modul findet sich im Quellcode.

e. Beendet wird das Programm mit <STRG>-<C>. Alle Ressourcen müssen dem Betriebssystem zurückgegeben werden, sonst kann es zu Problemen bei einer erneuten Ausführung der Simulation kommen. Also müssen Sie die entsprechenden cleanup Funktionen im Signalhandler zu `SIGINT` einhängen. Die mitgelieferte Datei. `mmanage.c` enthält hierfür bereits einzelne Teile des notwendigen Quellcodes.

4. Das Modul `vmem.h` definiert die Datenstruktur `struct vmem_struct` mit allen weiteren Strukturen und Konstanten. **Bitte arbeiten Sie vorab die mitgelieferte Dokumentation dieser Datenstrukturen durch.**

3. Aufgabenstellung

Schreiben Sie gemäß der oben spezifizierten Anwendung die Speicherverwaltung für `mmanage` und `vmaccess`. Der Code des Anwenderprogramms `vmappl` ist in TEAMs gegeben.

Hinweise:

- Zu Ihrer Unterstützung finden Sie neben dem Code des Anwendungsprogramms einige Hilfsfunktionen und Datenstrukturen im Code unter Teams. Die entsprechende DoxyGen Dokumentation liegt auch vor.

- In TEAMs finden Sie das shell script `run_all`. Es führt mehrere Simulationen durch und vergleicht die Simulationsergebnisse mit der Musterlösung (via `diff`). Das Skript führt die Simulation mit unterschiedlichen Sequenzen von Pseudozufallszahlen durch. Ein Vergleich mit der Musterlösung findet nur für den seed 2806 statt. Die anderen seeds sind standardmäßig auskommentiert um Rechenzeit zu sparen.
  Bitte schauen Sie sich das Skript im Detail an, da die dort implementierten Abläufe wesentlich sind. **Das `run_all` script muss im Rahmen der Abnahme erklärt werden.** In TEAMs finden Sie zudem ein Makefile für die Ausführung.

- Die Datei `run_all` erzeugt zu FIFO, CLOCK und Aging Varianten für Seitengrößen von 8, 16, 32 und 64 Integer Datenworten. Bei Aging wird zusätzlich ein Zeitintervall zum Abfragen der R-Flags gefordert. Dies wird durch den globalen Zähler `g_count` simuliert, der nach jedem Speicherzugriff (nicht davor!) inkrementiert wird.

4. Anforderungen an die Lösung

In Teams finden sich die Log Files für die drei Seitenersetzungsalgorithmen, wobei der Zufallszahlengenerator mit 2806 initialisiert wurde. Für die Abnahme wird überprüft, ob die Lösung hierfür die gleichen Log Files generiert. Es ist wichtig, dass Sie schon vor dem Praktikum sicherstellen, dass Ihre Log Files mit denen aus TEAMs übereinstimmen.

Beachten Sie hierzu folgende Tipps:

1. Die Log Files für die Seitenersetzungsalgorithmen FIFO und CLOCK sollten schnell mit Ihrer Lösung übereinstimmen. Erst wenn diese stabil laufen, sollten Sie das Aging Verfahren umsetzen.
2. Falls die Log Files zu Ihrem Agigng Verfahren mit den gegebenen Log Files übereinstimmen, dann müssen Sie anhand der Ausgaben Ihres Programms die korrekte Funktionsweise des Aging Algorithmus darstellen können.
3. Die Musterlösung und somit die Simulationsergebnisse für seed 2806 können fehlerhaft sein. Wenn Sie einen Fehler finden melden Sie mir dies bitte per Mail.

5. Hinweise/Troubleshoot

- Da ein OS unabhängiger sehr einfacher Zufallszahlengenerator in den Code integriert wurde, liefert die Simulation auf OS-X und Linux dieselben Ergebnisse. Damit können Sie die Aufgabe auf beiden Plattformen lösen.

- Schauen Sie sich die mitgelieferte Software in Ruhe an.

- Setzen Sie Ihre Lösung schrittweise um. Überprüfen Sie nach jedem Schritt, dass die Lösung das gewünschte Verhalten realisiert.

- Sie müssen auf der mitgelieferten Software aus diesem Semester aufbauen. Die Grundfunktion, dass Anwendung und Memory Manager separate Programme sind, muss erhalten bleiben. Dies gilt ebenfalls für die synchrone Kommunikation und das Shared Memory für vmem.h . Den Rest des Codes können Sie nach Belieben ändern, solange die Ergebnisse mit der Musterlösung (Vergleich der LogFiles) übereinstimmen.

- Zu Anfang werden wahrscheinlich die LogFiles des Aging Algorithmus von der Musterlösung abweichen. Ist dies der Fall, überprüfen Sie bitte folgende Punkte Ihrer Implementierung:

  - Wird Aging zum richtigen Zeitpunkt durchgeführt?
  - Führen Sie das Aging durch, nachdem der Page Fault behoben und das Reference Flag gesetzt wurde.
  - Können mehrere Pages aufgrund Ihres Alters ausgelagert werden (haben alle dasselbe Alter), lagern Sie die Page mit der höchsten Frame-Nummer aus.
  - Wird eine Seite eingelagert, setzen Sie den Age Zähler auf 0x80. Das ist wichtig, damit die frisch eingelagerte Seite nicht sofort wieder ausgelagert wird, wenn zwei Page Faults hintereinander auftreten, ohne das zwischenzeitlich das Age neu berechnet wurde.
- Als Anregung finden Sie im mitgelieferten Quellcode und in der Dokumentation die Signaturen der Funktionen der Musterlösung. Sie können natürlich eigene Konzepte verwirklichen.
- Alle Signale verwenden denselben Signalhandler. Der Quellcode zur Initialisierung und Installation der Signalhandler wird mitgeliefert.
- Falls Sie mit dem C-Debugger der Eclipse Umgebung arbeiten, müssen Sie - damit die in der Aufgabe verwendeten Signale nicht mit der Steuerung des Debuggers interferieren - der Debugger gemäß der Beschreibung der Datei README_DEBUGGER eingestellt werden.