Projekte SS24

Gruppenstärke pro Projekt, 2 Studierende

Die Bewertung des Projektes ist wie folgt verteilt:

- Implementierung C/C++ 50%, GUI 20%
  - Durchdachtes Konzept (Simulationsansatz, Abstraktionslevel, API und vorausschauendes Denken für mögliche Erweiterungen)
  - Präzise und saubere Implementierung (Coding-Style, schlanker Code, Fehlerfreiheit etc.)
  - Adäquat kommentierter Code (Doxygen)
- Dokumentation 20%
  - Verständliche Beschreibung des Projektes 5%
  - Bedienungsanleitung des Projektes Compilieren/Bedienen 5%
  - Kommentieren des Codes 5%
  - Beschreibung der Stuktur des Codes, evtl. UML, Doxygen,... 5%
- Präsentation 10%
  - Stil der Präsentation
  - Struktur / Inhalt der Folien
  - Umgang mit Fragen
  - Präsentationen finden in der zweiten Klausurphase statt. Termine werden später über eine Doodle-Umfrage mit Ihnen abgestimmt.

Durchzuführende Arbeiten bei allen Projekten

- Lesen der entsprechenden Kapitel im Buch 'Moderne Betriebssysteme' von A.Tannenbaum
- Planung der durchzuführenden Arbeiten, des Vorgehens (Meilenstein, insb. Erstellung eines Zeitplans)
- Aufteilung der Arbeit innerhalb der Gruppe
- Implementierung des selbst entwickelten Konzepts
- Anfertigung des Dokumentation, Erstellung der Präsentation, Präsentation der Arbeit

Beachten Sie, dass die Projekte bis spätestens 2 Wochen vor dem Präsentationstermin abgegeben werden müssen. Die Frist wird später bekannt gegeben

Projekt 1, Scheduling Simulator

Ziele
Innerhalb dieses Projekts soll eine Simulationsumgebung für Prozess Scheduler entwickelt und implementiert werden. Hierbei soll es möglich sein, beliebige Scheduling Algorithmen in die Umgebung einzubetten. Um dies zu erreichen soll eine Schnittstelle (API) definiert werden, mit welcher Entwickler ihre Scheduling Algorithmen einfach implementieren und evaluieren können. Der Fokus liegt hierbei auf einer vollständig vorhandenen Kernfunktionalität des Simulators.

Arbeitsbeschreibung
Wie bereits im Punkt 'Ziele' beschrieben soll innerhalb dieses Projekts eine Simulationsumgebung für Scheduling Algorithmen konzipiert und auch implementiert werden. Es wird somit ein Software-Konzept gesucht, welches Prozesse mit Hilfe des Discrete-Event (DES) Ansatzes[1] simuliert und einem Entwickler in diesem Kontext Zugang zum Planungsmechanismus des DES gewährt. Jedes Scheduling eines Prozesses im Simulator sei hierbei als eine Zuweisung von Rechenzeit einer CPU zu sehen. Hierbei wird von einer Single CPU Umgebung ausgegangen. Weiterhin sollen zur Veranschaulichung der Funktionalität des Simulators elementare Scheduling Algorithmen, die wir in der Vorlesung behandelt haben, implementiert werden. Die Algorithmen sind unten gelistet. Die zwei untengenannten Vorgehensweise des Scheduling von Threads müssen auch implementiert werden. Für das Scheduling von Threads können die Algorithmen des Prozess-Scheduling eingesetzt werden. Für dieses Projekt können Sie Ihre Lösung des 2 Praktikumszettel als Basis nehmen. Für dieses Projekt ist es unabdingbar, dass das entsprechende Kapitel aus dem Buch 'Moderne Betriebssysteme' vollständig verstanden wurde.

- Scheduling Strategie: Unterbrechendes
  - First Come First Served, Shortest Job First, Round Robin Scheduling, Round Robin mit Periorität
- Thread Scheduling
  - Verwaltung auf Kernebene
  - Verwaltung auf Benutzerebene

[1] https://omnetpp.org/
https://github.com/xMarkusSpringerx/discrete-event-simulation

Projekt 2, Paging Simulator

Ziele
Innerhalb dieses Projekts soll eine Simulationsumgebung für Paging Algorithmen entwickelt werden. Es soll eine Schnittstelle (API) definiert und implementiert werden mit deren Hilfe Paging Algorithmen einfach implementiert und evaluiert werden können. Der Fokus liegt hierbei auf einer vollständig vorhandenen Kernfunktionalität des Simulators.

Arbeitsbeschreibung
Wie bereits im Punkt 'Ziele' beschrieben soll innerhalb dieses Projekts eine Simulationsumgebung für Paging Algorithmen konzipiert und auch implementiert werden. Die Simulationsumgebung soll es erlauben Paging Algorithmen einfach implementieren und evaluieren zu können. Aspekte wie MMU und TLB müssen hierbei berücksichtigt werden. Weiterhin sollen zur Veranschaulichung der Funktionalität des Simulators elementare Paging Algorithmen, die wir in der Vorlesung behandelt haben, implementiert werden. Die Algorithmen sind unten gelistet. Zur Evaluation der Algorithmen muss natürlich eine Zeitbasis vorhanden sein, hierzu bietet sich der Ansatz von Diskrete-Ereignis Simulatoren (DES) an. Für dieses Projekt können Sie Ihre Lösung des 3 Praktikumszettel als Basis nehmen. Für dieses Projekt ist es unabdingbar, dass das entsprechende Kapitel aus dem Buch 'Moderne Betriebssysteme' vollständig verstanden wurde. Zum Entwickeln einer Simulationsumgebung wird jedoch noch Basiswissen über Diskrete-Ereignis Simulatoren (DES) benötigt[2].

Seitenersetzen Algorithmen

- Not Recently Used, First In First Out, Second Chance, Least Recently Used, Not Frequently Used (mit und Ohne Aging)

[2] https://omnetpp.org/
https://github.com/xMarkusSpringerx/discrete-event-simulation

Projekt 3, Dateisysteme

Ziele
Innerhalb dieses Projekts soll ein Framework zur Manipulation von Dateisystems implementiert werden. Es soll eine Schnittstelle (API) definiert und implementiert werden mit deren Hilfe Entwickler leicht bel. Änderungen innerhalb des Dateisystems vornehmen können.

Arbeitsbeschreibung Wie bereits im Punkt "Ziele"beschrieben soll innerhalb dieses Projekts ein Framework für Zugriffe auf das FAT/Inode-basiertes Dateisystem konzipiert und auch implementiert werden. Diese Framework soll einem Benutzer ermöglichen beliebige Operationen auf einem FAT/Inode Dateisystem vornehmen zu können. Für dieses Projekt können Sie Ihre Lösung des 4 Praktikumszettel als Basis nehmen. Für dieses Projekt ist es unabdingbar, dass das entsprechende Kapitel aus dem Buch 'Moderne Betriebssysteme' vollständig verstanden wurde. Zum konzipieren der Funktionalität des Frameworks ist es wichtig das FAT/Inode Dateisystem in technischer Hinsicht verstanden zu haben. Des Weiteren muss eine Simulation von einer CD-Laufwerk implementiert. Diese muss Datei-Lesen/Schreiben/Brennen auf die/ von der CD ermöglichen. Folgende Punkte können Ihnen bei der Gestaltung Ihres Projektes helfen:

- Aufteilung der Platte (Layout von Dateisystemen)
  - Master Boot Record; MBR
  - Zwei verschiedene Dateisysteme; FAT, I-Node basiert
- Speicher Belegung; Verkettete Listen
- Verzeichnisse; FAT und I-Nodes
- Defragmentierung
- Statistische Anzeige der Plattenbelegung (Fragmentieungsstatus)
- Datei auf den / von dem CD-ROM schreiben / lesen

Projekt 4, Deadlocks

Ziele
Innerhalb dieses Projekts soll eine Simulationsumgebung für Deadlocks Algorithmen entwickelt werden. Es soll eine Schnittstelle (API) definiert und implementiert werden mit deren Hilfe Deadlocks Algorithmen einfach implementiert und evaluiert werden können. Der Fokus liegt hierbei auf einer vollständig vorhandenen Kernfunktionalität des Simulators.

Arbeitsbeschreibung
Wie bereits im Punkt 'Ziele' beschrieben soll innerhalb dieses Projekts eine Simulationsumgebung für Deadlocks Algorithmen konzipiert und auch implementiert werden. Die Simulationsumgebung soll es erlauben Deadlocks Algorithmen einfach implementieren und evaluieren zu können. Aspekte wie Erkennung, Vemeidung und Verhinderung von Deadlocks müssen hierbei berücksichtigt werden. Weiterhin sollen zur Veranschaulichung der Funktionalität des Simulators elementare Deadlocks Algorithmen, die wir in der Vorlesung behandelt haben, implementiert werden. Die Algorithmen sind unten gelistet. Zur Evaluation der Algorithmen muss natürlich eine Zeitbasis vorhanden sein, hierzu bietet sich der Ansatz von Diskrete-Ereignis Simulatoren (DES)[3] an. Für dieses Projekt ist es unabdingbar, dass das entsprechende Kapitel aus dem Buch 'Moderne Betriebssysteme' vollständig verstanden wurde. Folgende Punkte können Ihnen bei der Gestaltung Ihres Projektes helfen:

- Erkennen und Beheben von Deadlocks einer Ressource mehreren Ressourcen pro Typ
  - Behebung durch Unterbrechung
  - Behebung durch teilweise Wiederholung (Rollback)
  - Behebung durch Prozessabbruch
- Verhinderung von Deadlocks
  - Konzept der sicheren Zustände
  - Bankier-Algorithmus für eine einzelne Ressource
  - Bankier-Algorithmus für mehrere Ressourcen
- Vermeidung von Deadlocks
  - Alle Ressourcen werden zu Beginn eines Prozesses angefordert (Eliminierung von "Hold and Wait")
  - Priorisierung der Prozesse (Eliminierung von "No Preemption")
  - Nummerierung der Ressourcen, nur Aufsteigender Zugriff ist erlaubt (Eliminierung von "Circular Wait")

[3] https://omnetpp.org/
https://github.com/xMarkusSpringerx/discrete-event-simulation