Eine Zusammenstellung aus Prüfungsprotokollen
bei Professor Schlageter

Praktische Lernkarten zum Ausschneiden, Zusammenkleben und Sammeln :-) zur Prüfungsvorbereitung Diplomprüfung

Betriebssysteme

Thomas Schwarze
Thomas.Schwarze@FernUni-Hagen.de

20. Dezember 2003

Diese Arbeit wurde mit dem Textsystem LaTeX erstellt.

Was ist ein Prozess?

Welche Ziele werden bei der Prozessorvergabe verfolgt?
Was soll maximiert bzw. minimiert werden?

Welches Scheduling-Verfahren/ Prozessorzuteilungsalgorithmen wendet man in Batch-Systemen an?

Wann findet denn bei Batch-Systemen ein Prozesswechsel statt?

Wie sieht das Scheduling-Verfahren bei Dialog-Systemen aus?

Was passiert bei Dialog-Systemen, wenn ein sehr langer Prozess ankommt?

Warum ist das Feedback-Scheduling sinnvoll?

Was ist zu tun, wenn Batch- und Dialogsysteme gleichzeitig in einem System betrieben werden?

- maximale Fairness („kein Verhungern“)
- minimale Antwortzeit (aus Sicht des Benutzers)
- maximaler Durchsatz (viele Aufträge pro Zeit)
- minimale Durchlaufzeit (kurze Abarbeitung eines Programms bei Batch-Programmen)

Ein in der Ausführung befindliches Programm. Ein Programm ist statisch, ein Prozess ist dynamisch.

Bei Beendigung eines Prozesses oder beim Warten auf ein Ereignis.

Non-preemptive wie

- FCFS (First Come First Served), Nachteil: Kurze Prozesse haben eventuell lange Wartezeit
- SJF (Shortest Job First), minimale Antwortzeit, Nachteil: Lange Prozesse können verhungern
- Priority Scheduling, Nachteil: Unfair bei statischer Prioritätenvergabe

Einsatz von rechenzeitabhängigen Feedback Scheduling, Prozess erhält hohe Priorität mit kleiner Zeitscheibe. Wurde Quantum verbraucht, dann niedrigere Priorität mit doppelter Zeitscheibe und umgekehrt. Lange Laufzeiten erhalten seltener aber längere Zeitscheiben und vice versa. Nachteil: Hoher Verwaltungsaufwand.

Pre-emptive wie

- Round Robin, wie FCFS mit Zeitscheibe, aktueller Prozess kommt an das Ende der Queue
- SJF (Shortest Job First), wie Non-preemptive aber mit Unterbrechung bei neuen Prozessen
- Priority Scheduling

Kombination der Scheduling Verfahren zum Feedback Scheduling oder Einteilung der Prozesse in Klassen und Einsatz einer Multiple Queue.

Prozesswechsel sind sehr teuer, daher muss die zur Verfügung stehende Prozessorzeit (das Quantum für jeden Prozess) möglichst optimal ausgenutzt werden.
Anpassung der Zeitscheiben an CPU-Burst (Zeit, in der Prozess die CPU behalten möchte).

Beschreiben Sie die Unterbrechung eines Prozesses.

Wie kommen Prozesse in Batch-Systemen mit größerem Zeitbedarf zu ihrem Recht?

Erklären Sie die Round Robin Strategie.

Welche Klassen von Zuteilungsverfahren gibt es?
Wann wendet man welche an?
Welche Möglichkeiten gibt es, die Prozessorzuteilung zu beeinflussen?

Was ist eine Arbeitsmenge?

Wie bestimmt man die Arbeitsmenge?
Wie stellt man fest, welche Rahmen zur Arbeitsmenge gehören?

Welche anderen Seitenzuweisungsstrategien außer der Arbeitsmengenstrategie gibt es noch?

Welche Seitenzuweisungsstrategie ist vorteilhafter?

Entweder FCFS (First Come First Served) oder statische bzw. dynamische Prioritätenvergabe (Priority Scheduling)

Die Unterbrechung erfolgt in Batch-Systemen non-preemptive und in Dialog-Systemen preemptive.
Umschaltung erfolgt bei:

- Prozess will I/O oder wartet auf Ereignis
- Prozess gibt CPU freiwillig ab oder ist fertig
- Zeitscheibe ist abgelaufen
- Neuer Prozess wird erzeugt
- Ereignis für blockierten Prozess tritt ein

Non-Preemptive und Preemptive Zuteilungsverfahren. Erste werden fast ausschließlich in Batch-, die anderen in Dialog-Systemen eingesetzt. Die Beeinflussung der Zuteilung geschieht durch den Einsatz unterschiedlicher Scheduling-Verfahren und innerhalb dieser durch unterschiedliche Maßnahmen (Prioritätenfestsetzung, Größe der Zeitscheibe, Einteilung in Klassen (System-, Dialog- und Hintergrundprozesse)).

Wie FCFS (First Come First Served) mit Zeitscheiben (Quantum). Hat ein rechnender Prozess nach Ende der Zeitscheibe CPU nicht freigegeben, unterbricht das Betriebssystem und der Prozess kommt an das Ende der Warteschlange.

Mit Zugriffbits in Schieberegistern.
Indem alle Seiten erfasst werden, auf die in „letzter Zeit“ von dem Prozess zugegriffen wurde.

Menge der Seiten des virtuellen Speichers eines Prozesses, zu denen „letzter Zeit “ zugegriffen worden ist; approximiert die Menge der „aktuell benötigten“ Seiten.

Nachteil der Seitenfehlerhäufigkeitsstrategie ist, dass der Schaden erst eintreten muss, bevor gegengesteuert wird. Man kann aber Arbeitsmengenstrategie durch Seitenfehlerhäufigkeitsstrategie kontrollieren und Konsequenzen ziehen.

Seitenfehlerhäufigkeitsstrategie oder auch Seitenfehlerrate.

Welche Seitenauslagerungsstrategien gibt es?

Was wäre die optimale Seitenauslagerungsstrategie?
Welche Seiten wären auszulagern?

Warum treibt man bei der Hauptspeicherverwaltung solchen Aufwand?

Wie bestimmt der Systemadministrator das Δ der Zeitspanne bei der Approximation der Arbeitsmenge?

Wie erkennt man, dass die Arbeitsmenge erreicht ist?

Welche Seitenzuteilungsstrategien gibt es?
Wie funktionieren diese?

Wie werden mehrere Prozesse im Speicher verwaltet?

Wie läuft das mit der Seitenein- und Seitenauslagerung?

Auslagern der Seite, die erst am Weitesten in der Zukunft wieder benutzt wird.

- LRU (Least Recently Used), aufgrund Lokalitätsannahme Auslagerung der Seiten, auf die am Längsten nicht zugegriffen wurde.
- Zugriffbits, ähnlich wie LRU aber Beobachtung über größeren Zeitraum (z.B. 1000 Zugriffe) und Sicherung in Schieberegister (z.B. 8 Bit für 8 Zeiträume). Keine kleinteilige Unterscheidung innerhalb des Zeitraums.
- FIFO, älteste Seite fliegt raus. Erweiterung mit „last chance“ in extra Liste. Wenn doch noch ein Zugriff erfolgt, dann wieder ans FIFO-Ende.

Da Auslagerung auf die Festplatte erfolgt, kann bei schneller Festplatte kleinere Arbeitsmenge gewählt und häufigeres Laden bei Seitenfehlern aus dem paging-Bereich in Kauf genommen werden. Wählt man das Fenster Δ zu groß, wird die Arbeitsmenge unnötig groß und die Zahl möglicher paraller Prozesse sinkt oder es kommt zur Systemüberlastung (Trashing = Behördenverhalten: Selbstbeschäftigung ohne Output) wegen dauernder Ein- und Auslagerung.

Zuteilung der erforderlichen Arbeitsmenge für jeden Prozess.

- Statische Zuteilung von Seitenrahmen an Prozesse
- Arbeitsmengenstrategie
- Seitenfehlerhäufigkeitsstrategie

Seitenfehlerrate sinkt, Kontrolle durch Seitenfehlerhäufigkeitsstrategie.

Benötigt ein Prozess weitere Seitenrahmen für zuladbare Programmteile oder für Daten (Seitenfehler), werden dem Prozess neue Seitenrahmen zugewiesen. Existieren keine freie Seitenrahmen mehr, müssen belegte, fremde Seiten ausgelagert werden. Will der Prozess wieder auf ausgelagerten Seite zugreifen, muss diese Seite erst eingelagert werden.

Durch Zuteilung von Seitenrahmen.

Was geschieht, wenn alle Seiten bereits vergeben sind?

Was kann man gegen Systemüberlastung wegen „Trashing“ bzw. Seitenflattern tun?

Welche Seiten gehören zur Arbeitsmenge?

Welche Strategien verwendet man bei dynamischen Systemen, um die Größe der Arbeitsmenge anzupassen?

Wie stellt man fest, ob Seiten hinzugefügt werden müssen?

Wie läuft die Seitenverdrängung bei der Arbeitsmengenstrategie ab?

Was beeinflusst die Größe der Arbeitsmenge der Prozesse am meisten?

Wird die Arbeitsmenge bei hoher Zugriffsgeschwindigkeit der Festplatten größer oder kleiner?

Die Zahl der aktiven Prozesse muss verringern werden, indem einige Prozesse z.B. mit geringer Priorität vorübergehend völlig stillgelegt und ihre belegten Seitenrahmen freigegeben werden, da die CPU sonst nur noch mit Ein- und Auslagerungen beschäftigt ist.

Fremde Seiten aus anderen Arbeitsmengen werden ausgelagert und dem Prozess als neue Rahmen zugewiesen.

Die Arbeitsmengenstrategie als Seitenzuweisungsstraegie.

- Bei jedem Seitenfehler wird einem Prozess ein weiterer Seitenrahmen zugewiesen.
- Wenn eine Seite aus der Arbeitsmenge herausfällt, also zu lange nicht mehr benutzt wurde, wird dem Prozess ein Seitenrahmen entzogen. Dazu werden sinnvoller Weise nur LRU bzw. approximiert Zugriffsbits genommen.

Alle Seiten, die „in letzter Zeit“ benutzt worden sind.

Seiten, auf die lange nicht mehr zugegriffen wurden, gelten nicht mehr als lokal und fallen damit aus der Arbeitsmenge heraus. Die damit belegten Seitenrahmen werden dem Prozess entzogen.

Wenn ein Seitenfehler erzeugt wird (Zugriff auf nicht geladene Seite).

Kleiner, da die Wiedereinlagerungskosten kleiner sind.

- Die Zugriffsgeschwindigkeit der Festplatten (will Schlageter hören).
- Die Größe des Arbeitsspeichers im Verhältnis zur Summe der Arbeitsmengen aller Prozesse.
- Die Größe Δ des Zeitfensters.

Wann weiß man, ob die Seitenfehlerrate zu hoch ist?

Wie muss sich das Betriebssystem verhalten, wenn die Seitenfehlerrate zu groß wird?

Was bezweckt man mit der Realisierung von virtuellem Speicher?

Woher weiß das System, dass es neue Seitenrahmen erhalten kann?

Was macht man, wenn alle Seitenrahmen von Prozessen belegt sind und ein Prozess benötigt noch Rahmen?

Wie kann man ohne LRU (least recently used) feststellen, welche Seiten nicht mehr zur Arbeitsmenge gehören?

Wovon wird die Seitenfehlerrate beeinflusst?

Welche Synchronisationsmethoden gibt es?

Es muss die Zahl der aktiven Prozesse verringern, indem einige Prozesse z.B. mit geringer Priorität vorübergehend völlig stillgelegt und ihre belegten Seitenrahmen freigegeben werden, da die CPU sonst nur noch mit Ein- und Auslagerungen beschäftigt ist.

Wenn die Summe der Arbeitsmengen aller parallel laufender Prozesse größer ist, als der verfügbare physische Hauptspeicher.

Wenn die Seitenfehlerrate sinkt und wenn die Summe der Arbeitsmengen aller parallel laufender Prozesse kleiner ist, als der verfügbare physische Hauptspeicher.

Zuteilung der erforderlichen Arbeitsmenge für jeden Prozess.

Indem Zugriffe in Zugriffbits (in Schieberegistern) festgehalten werden.

Verringerung der Zahl der aktiven Prozesse, indem Prozesse mit geringer Priorität vorübergehend völlig stillgelegt und ihre belegten Seitenrahmen freigegeben werden.

- Allgemeine Synchronisationsvariable
- Semaphore
- Nachrichtenaustausch
- Monitor

- Die Zugriffsgeschwindigkeit der Festplatten (will Schlageter hören).
- Die Größe des Arbeitsspeichers im Verhältnis zur Summe der Arbeitsmengen aller Prozesse.
- Die Größe Δ des Zeitfensters.

Es gibt Semaphore, die einen Wert > 1 annehmen. Nennen Sie ein Beispiel.

Was sind Monitore? Wo ist der Unterschied zu Semaphoren?

Wie sähen Monitore bei Festplatten aus?

Wir haben im System gemeinsamen Zugriff auf Betriebsmittel. Welche Probleme sind zu beachten und welche Lösungsmöglichkeiten gibt es?

Wie funktionieren Semaphore?

Was bedeutet es, wenn Semaphore Werte > 1 annehmen können?

Welche Datenstrukturen sind im Monitor enthalten?

Was ist die P-Operation im Semaphorenmodell von Dijkstra und wie wird sie angewandt?

Eine Menge von Datenstrukturen und Prozeduren, die der Synchronisation von Prozessen dienen und die als Betriebsmittel betrachtet mehreren Prozessen zur Verfügung stehen, aber immer nur von einem zur Zeit benutzt werden können.
Der Unterschied zu Semaphoren ist die „Umkapselung“ von Betriebsmitteln ähnlich wie Klassen der objektorientierten Programmierung.

Beim Erzeuger / Verbraucher-Problem wird die Menge der erzeugten Produkte, die noch nicht verbraucht wurden, im Semaphoreninhalt gespeichert.

Zugriffe auf gemeinsam genutzte Betriebsmittel stellen kritische Abschnitte in Prozessen dar, die miteinander synchronisiert werden müssen. Dabei darf es weder zum gleichzeitigen Zugriff kommen noch darf ein Deadlock eintreten. Zur Lösung gibt es Synchronisationsvariablen, Nachrichten, Semaphore und Monitore.

Monitore bei Festplatten sind die Kontroller, über die sämtliche Zugriffe gehen müssen. Eine Umgehung der Kontroller mit direktem Zugriff auf die Laufwerkselektronik ist ausgeschlossen. In dem Kontroller sind abschließend alle Prozeduren vereint, die für die Zusammenarbeit einer Festplatte mit dem Betriebssystem notwendig sind.

Dann können mehr als ein Prozess gleichzeitig in den kritischen Abschnitt eintreten bzw. das Betriebsmittel nutzen.

Spezielle Synchronisationsvariable, die nur über P- und V-Operatoren getestet bzw. verändert werden darf.
Der P-Operator dekrementiert dabei den Semaphor, wenn er größer 0 ist. Andernfalls wartet der P-Operator solange, bis der Semaphor größer 0 wird.
Der V-Operator inkrementiert den Semaphor.
Es gibt binäre und allgemeine Semaphore.

Der P-Operator dekrementiert den Semaphor, wenn er größer 0 ist. Andernfalls wartet der P-Operator solange, bis der Semaphor größer 0 wird.
Angewandt wird der P-Operator, indem er als Funktion oder Prozedur aufgerufen wird. Eine Realisierungsmöglichkeit ist die durch einen Monitor.

Die mindesten Datenstrukturen sind Prozeduren für das Belegen und Freigeben des Betriebsmittels mit der dazu notwendigen internen Variablen zur Kennzeichnung, ob gerade ein Prozess das Betriebsmittel belegt hat. Darüber hinaus sind beliebige weitere Betriebsmittel abhängige Daten denkbar und sinnvoll.

Von wem wird ein durch den P-Operator schlafen gelegter Prozess wieder aufgeweckt, wenn der Semaphor durch einen anderen Prozess wieder inkrementiert wurde?

	Durch den Dispatcher.