Betriebssysteme

Tutorium 4

Philipp Kirchhofer
philipp.kirchhofer@student.kit.edu
http://www.stud.uni-karlsruhe.de/~uxbtt/

Lehrstuhl Systemarchitektur
Universität Karlsruhe (TH)

18. November 2009

Was machen wir heute?

1. POSIX
2. Tutorien Übungsblatt
   - More on Processes in Unix
   - Threads
   - Scheduling Basics

POSIX

Portable Operating System Interface [for Unix] (POSIX)

Ab 1985 entwickelter Standard, legt die Schnittstellen zwischen Betriebssystem und Anwendungsprogrammen fest.

Bestandteile

1. Konventionen
2. System-Schnittstellen
3. Kommandozeileninterpreter (Shell) und Hilfsprogramme
4. Erklärungen

Nützliche Seiten

http://linux.die.net/man/ Linux man pages
http://www.opengroup.org/ Open Group: System Interfaces TOC
http://technet.microsoft.com/ Microsoft Subsystem for UNIX-based Applications

Tutorien Übungsblatt - More on Processes in Unix

Frage 4.1.a

Erstelle ein C-Programm mit der Funktionalität von *ls* | *less*.

Antwort

Demo *pipe.c*

Pipe erstellen

```
int pipe(int pipefd[2]);
```

FDs duplizieren

```
int dup2(int fd1, int fd2);
```

Programmcode laden und ausführen

```
int execlp(const char *file, const char* arg0, ..., NULL);
```


Frage 4.1.b

Vergleiche *fork* mit dem *CreateProcess* Systemaufruf der Windows API. Welche Unterschiede gibt es?

Antwort

CreateProcess vereinigt die Funktionalität von *fork* und *execve*. Dieses Vorgehen spart einen Systemaufruf, aber verringert auch die Flexibilität.

Durch die Kapselung der Prozesserzeugung (*fork*) und Ausführung eines neuen Programmcodes (*execve*) kann der Programmkontext des Kindprozesses einfach geändert werden.
Unter Windows muss hierzu eine spezielle Datenstruktur an den Systemaufruf übergeben werden.

Frage 4.2.a

Für welche Anwendungen sind Threads vorteilhaft?

Antwort

Computerspiele

- Thread für Grafik (Rendering)
- Thread für KI
- Thread für Netzwerk
- Thread für Physiksimulation

Webserver
Ein Thread verteilt ankommende Anfragen auf verschiedene Arbeiter-Threads. Diese verarbeiten die jeweilige Anfrage und geben das gewünschte Ergebnis zurück.

Bildverarbeitung
Ein Bild wird in mehrere Teilbereiche eingeteilt. Jeder Teilbereich wird von einem anderen Thread bearbeitet.

Frage 4.2.b

Welche Vorteile ergeben sich bei der Verwendung einer Anwendung mit mehreren Threads im Vergleich zu einer Anwendung, die verschiedene Instanzen von sich erstellt (fork)?

Antwort

Geringerer Overhead
Die Erzeugung eines Threads ist schneller als die Erzeugung eines Prozesses.

Datenaustausch
Threads können sowohl einfach auf gleichen Daten arbeiten als auch über gemeinsame Speicherbereiche kommunizieren.

Frage 4.2.c

Beschreibe die drei Thread-Modelle und deren jeweilige Vor- und Nachteile:

- Many-to-One
- One-to-One
- Many-to-Many

Many-to-One

- ⊕ Kein Kernaufruf bei Threadwechsel
- ⊖ Blockierender Systemaufruf blockiert Prozess
- ⊖ Keine Verteilung der Threads auf verschiedene Prozessoren

One-to-One

- ⊕ Blockierender Systemaufruf blockiert nur den aufrufenden Thread
- ⊕ Verteilung der Threads auf verschiedenene Prozessoren möglich
- ⊖ Overhead im Vergleich zu Many-to-One Modell höher

Many-to-Many

- ⊕ Kombiniert Flexibilität des One-to-One Modells mit dem geringeren Overhead des Many-to-One Modells
- ⊖ Kommunikation zwischen Anwendungs- und Kernel Scheduler nötig

Frage 4.2.d

Ein Prozess mit mehreren Threads ruft *fork* unter Linux auf. Was passiert?

Antwort

fork erstellt unter Linux nur eine Kopie des aktuellen Threads. Der Kind-Prozess enthält also nur einen einzigen Thread.

Frage 4.3.a

Was ist der Zweck von „scheduling“?

Antwort

Scheduling ist die Erstellung eines Ablaufplanes, der Prozessen die jeweils benötigten Ressourcen zuweist. Normalerweise versucht man mit Scheduling bestimmte Kriterien (z.B. Ressourcenauslastung) zu optimieren.

Frage 4.3.b

Welche Kriterien können für eine Bestimmung der Qualität eines Ablaufplanes herangezogen werden?

Antwort

Prozessorauslastung Anteil der Zeit, in der der Prozessor nichts zu rechnen hat

Durchsatz Anzahl der Prozesse/Threads, die in einer bestimmten Zeit abgeschlossen wurden

Verweilzeit Zeit vom Starten eines Prozesses/Threads bis zu seiner Beendigung

Wartezeit Zeit, die ein Prozess/Thread in der Warteschlange verweilt

Antwortzeit Zeit vom Starten eines Prozesses/Threads bis zur ersten Antwort

Frage 4.3.c

Welche der Kriterien sollen maximiert, welche minimiert werden?

Antwort

Maximierung

- Prozessorauslastung
- Durchsatz

Minimierung

- Verweilzeit
- Wartezeit
- Antwortzeit

Frage 4.3.d

Prozess	Durchlaufzeit	Startzeit
P1	7	2
P2	3	0
P3	1	7

Gegeben seien die Prozesse P1, P2 und P3.
Zeichne ein Gantt Diagramm eines FIFO-Schedule. Berechne die durchschnittliche Wartezeit und die durchschnittliche Verweilzeit.

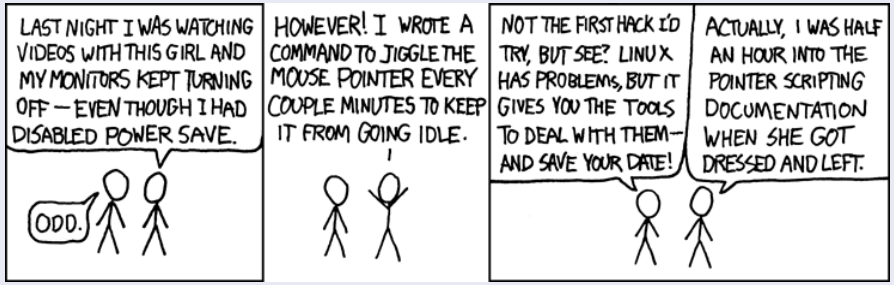
Antwort

Antwort

Durchschnittliche Wartezeit: $T^w_{\text{avg}} = \frac{0+1+3}{3} = \frac{4}{3}$

Durchschnittliche Verweilzeit: $T^t_{\text{avg}} = \frac{3+8+4}{3} = \frac{15}{3} = 5$
Hinweis: Verweilzeit = Wartezeit + Bearbeitungszeit

Fragen & Kommentare?



xkcd: Command Line Fu