SMP, Threading und die klassische IPC

Matthias Sattler
Unix-AG, Universität Kaiserslautern
sattler@unix-ag.uni-kl.de

17. Mai 1997

1 IPC

Mein Vortrag behandelt die „klassische IPC“ (InterProcess Communication) das Threading und das SMP (Symmetric MultiProcessing). Da dieser Text meine Vortragsfolien nur ergänzt, ist es vorteilhaft, diese während des Lesens zur Hand zu haben. Mir ist mehr daran gelegen, einen konzeptuellen Überblick zu geben, anstatt mit vielen Details aufzuwarten, die man leicht nachlesen kann.

Mit meinem unkonventionellen Ansatz zun den Themen IPC und Threads in diesem Vortrag versuche ich dem Leser nahe zu bringen, daß das Threading auch als eine Form der IPC gesehen werden kann. Dafür definiere ich einige Begriffe ein wenig abstrakter als allgemein üblich, um die konzeptionellen Gemeinsamkeiten besser herausstellen zu können.

Ich definiere einen „Execution Context“ als die Verbindung von Rechenzeit mit einem Stackbereich. Dies ist natürlich noch nicht genug um sinnvoll damit zu rechnen, allerdings sind dies die beiden Dinge, die zum Ablauf einer Codesequenz exklusiv für diese Ausführung nötig sind.

Ein Process hat alle nötigen sonstigen Resourcen alleine für sich, während ein Thread viele Resourcen mit anderen Threads teilt. Beiden gemein ist, daß sie einen jeweils eigenen Execution Context besitzen. Wenn man nun sowohl den Process als auch den Thread als Execution Contexte in verschiedenen Ausprägungen betrachtet stellen sowohl die klassische IPC als auch das Threading lediglich eine Kommunikationsmöglichkeit zwischen Execution Contexten dar.

Linux stellt sowohl die meisten klassischen IPC-Möglichkeiten (Pipes, Named Pipes, Shared Memory, Signals und Semaphores), als auch Threads dem Programmierer bereit, wobei Signals und Semaphores hauptsächlich der IPC Synchronisation dienen. Sowohl der aktuelle stabile Betriebssystemkern wie auch die libc unterstützen Execution Contexte in den verschiedensten Ausprägungen, was also den Threadsupport beinhaltet. Es existieren allerdings noch Systemteile (z. B. manche anderen Bibliotheken) die Threads nicht voll unterstützen, was vor deren Benutzung bedacht werden muß.

Wie man aus der nun folgenden Abbildung erkennen kann, benötigen sowohl Pipes als auch Named Pipes und Threads bei kleinen Projekten einen geringen Entwicklungsbedarf. Das Verhalten von Pipes und Named Pipes wird aber schon bei mittelgroßen Projekten recht schlecht, während das Threading mit dem Shared Memory bei auch großen Projekten vom Aufwand her fast mithalten kann. Dieser hat allerdings einen sehr großen Grundaufwand, der sich bei kleinen Projekten stark bemerkbar macht.

Nicht betrachtet habe ich die IPC mittels „Streams“ die von Unix System V kommt, obwohl Linux diese zur Verfügung stellen kann. Sie werden auch nur selten benutzt.

2 SMP

Der aktuelle, stabile Linux Betriebssystemkern unterstützt eine einfache Art des SMP, bei der sich immer nur eine CPU gleichzeitig im Kernelbereich aufhalten kann. Dies hat zur Folge, daß der Gewinn durch mehrere CPUs bei vorwiegend I/O-intensiven Anwendungen recht klein ist. Die Entwicktung zielt zur Zeit in diese Richtung, um allen CPUs gleichzeitig Zugang zum Kernel zu gewähren.