GNU/Hurd

... ein Mach basiertes Multi Server Betriebssystem

Manuel Gorius

16. September 2004

Übersicht

1. Wissenswertes zu Hurd
2. Motivationen und Ziele
3. Mach Microkernel
4. Single Server-Ansatz vs. Multi Server-Ansatz
5. Ausgewählte Server und Funktionen

1. Wissenswertes zu Hurd

Hurd? Was ist das?

- Ein auf GNU Mach Microkern aufbauendes Multi-Server Betriebssystem
- Ausschliesslich freie Software
- "GNU/FSF project's replacement for the Unix-Kernel"

Historisches

- 1991: Gründung des Hurd-Projektes durch Thomas Bushnell
- 1994: Hurd wird erstmalig gebootet
- 1997: Release der Version 0.2 von Hurd
- 1998: Erstellen eines Debian hurd-i386 Archives

2. Motivationen und Ziele

GNU/Hurd - Motivationen

Design-Probleme durch den monolithischen Systemkern

- Alle Systemfunktionen im Kernel
- Erweiterungen und Änderungen benötigen Zugriff auf den Kernel
- Beispiel: Mounten von Dateisystemen

GNU/Hurd - Ziele

- Modernes, objektorientiertes Design
- Erweiterbar
- Stabil
- Kompatibel

3. Mach Microkernel

Mach Microkernel

- Entstand 1985 an der Carnegie Mellon University
- Zunächst unter Verwendung des BSD UNIX Kernel entwickelt
- Bis Version 2.5 anwachsender Funktionsumfang im Kernel
- Version 3.0: Versuch, den Kern wieder zu verkleinern

Mach Microkernel

6 Abstraktionen des Mach

- Task
- Thread
- Port
- Message
- Memory Object
- Geräte

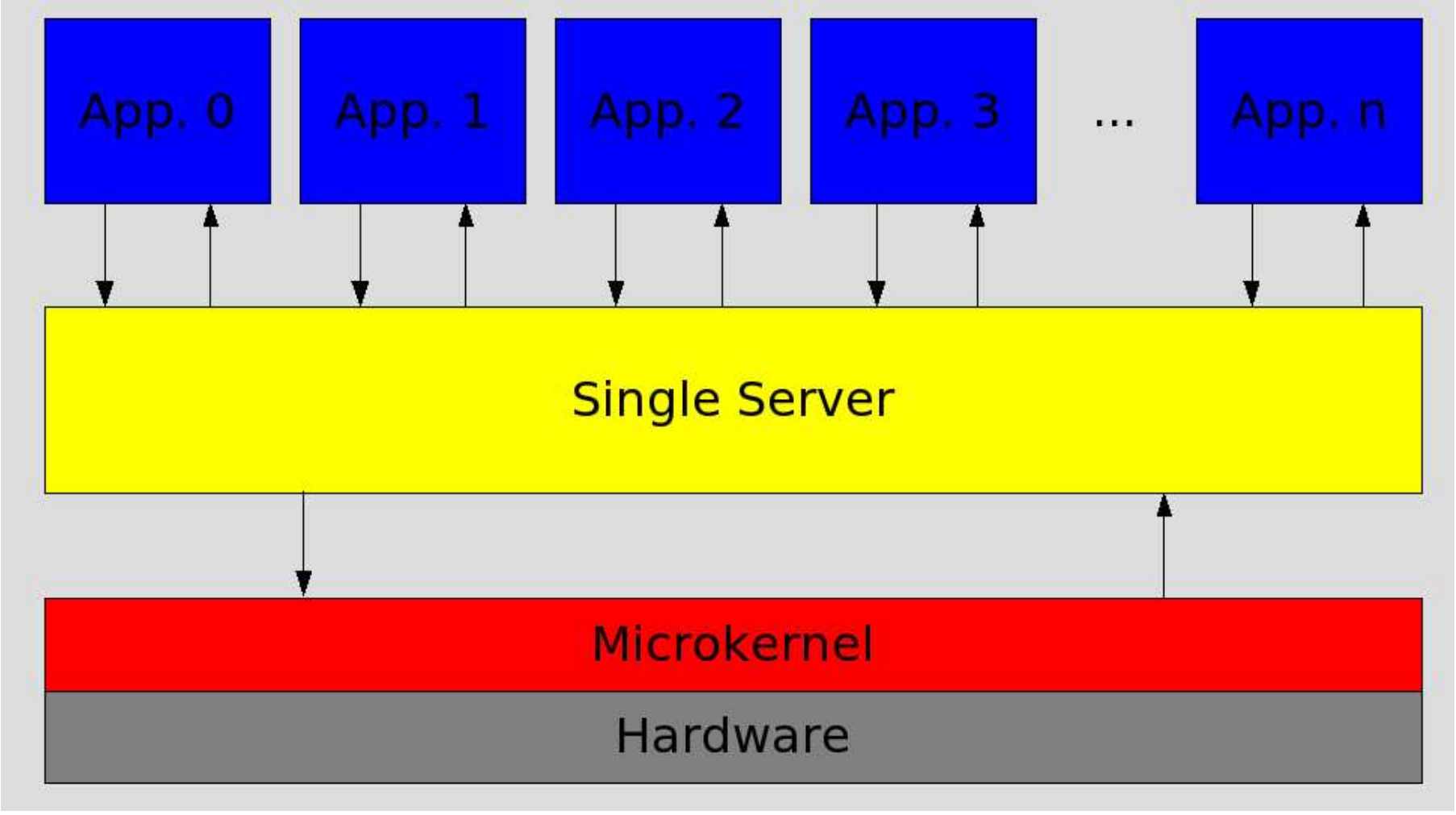
4. Single Server vs. Multi Server

Single Server vs. Multi Server

Single Server:

- Einzelner Task implementiert die Funktionalität des Betriebssystems
- Vergleichbar zum monolithischen Kernel
- Modifizierter Kernel eines monolithischen BS als Server
- Prozesse kommunizieren mit diesem einen Server
- Beispiel: Linux on L4 (TU Dresden)

Single Server vs. Multi Server

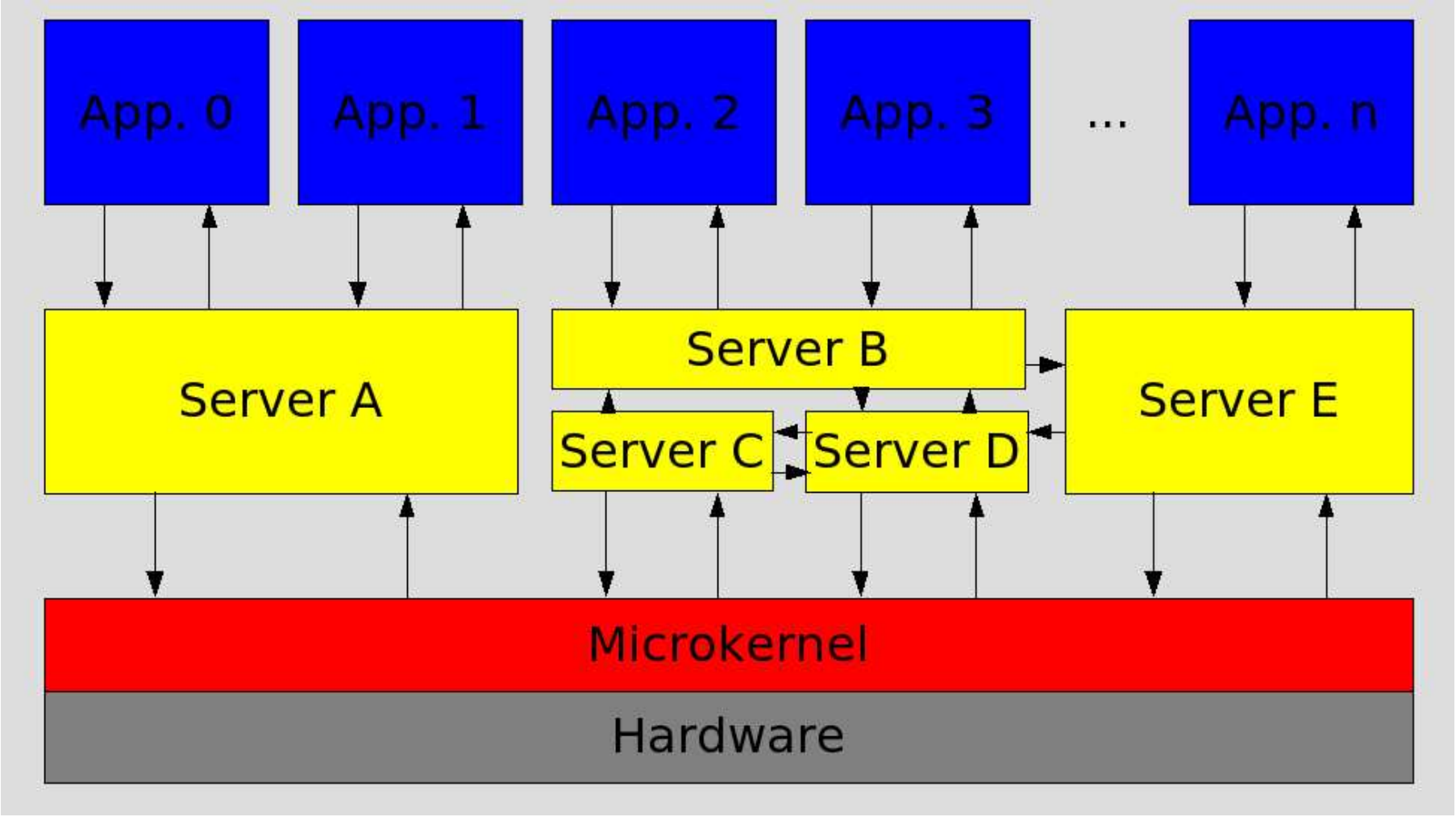


Single Server vs. Multi Server

Multi Server:

- Mehrere Tasks
- Jeder Server als kleine, wohldefinierte Einheit des gesamten Systems
- Klar abgegrenzte Verantwortlichkeiten jedes Servers

Single Server vs. Multi Server



Single Server vs. Multi Server

Vergleich: Single Server, Multi Server

- Keine Isolierung <-> Ausführungsintegrität der Server
- Kurze Kommunikationswege <-> aufwendige IPC
- Einfachere Konstruktion <-> aufwendige Auftrennung in Module

Single Server vs. Multi Server

Ergebnis für Hurd: Vorteile des Multi Servers durch

- ... statische Eigenschaften (Entwicklung)
  - Modularisierung
  - Sämtliche Teile getrennt entwickelbar
  - Reduzierte Komplexität

- ... dynamische Eigenschaften (Laufzeit)
  - Klare Trennung der Verantwortlichkeiten
  - Bessere Ausfalltoleranz
  - Bessere Erweiterungstoleranz

5. Ausgewählte Server u. Funktionen

IPC in Hurd

Mach Ports: "message queues",
Einweg-Kommunikationskanäle

- Rechtevergabe für Ports: receive, send oder send-once
- receive-Recht für genau einen Prozess
- send-Recht für mehrere Prozesse

IPC in Hurd

Vergabe von Ports im Mach

- Nameserver Verwaltet eine Liste von registrierten Servern
- Vergabe von Ports an alle registrierten Server
- Nameserver-Port durch Mach gegeben
- Bei Anfrage auf einen Port: lookup in der Serverliste

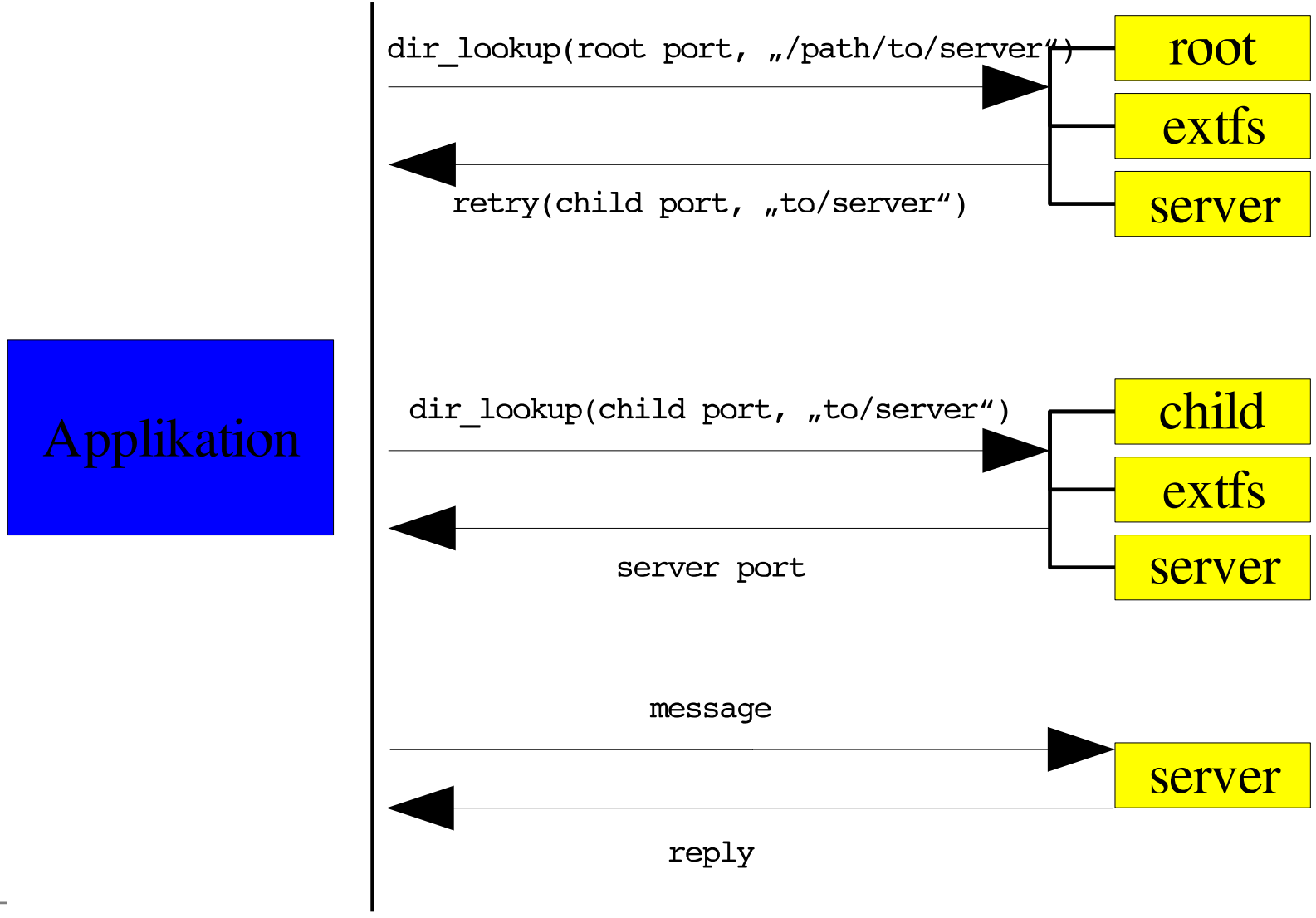
IPC in Hurd

Lösung in Hurd

- Dateisystem als Server-Namespace
- Root-Verzeichnis Port in jeder Task
- Auflösung des Pfadnames mit hurd_file_name_lookup (C-library)
- Baumartiges Verzeichnis

IPC in Hurd

Funktion file_name_lookup()



Translator-Mechanismus

Üblicher Dateizugriff in Hurd

- Jedes Verzeichnis gehört einem Server an.
- Beim Öffnen einer Datei erzeugt der Server einen Port, verknüpft mit der Datei.
- Der Port wird an das öffnende Programm zurückgegeben.

Translator-Mechanismus

Dateizugriff mit Translator

- Statt Port zurückzugeben startet der Server ein Translator-Programm.
- Das Translator-Programm erhält den Port auf den Inhalt der Datei.
- Das Translator-Programm gibt einen Port an das aufrufende Programm zurück.
- Beispiele: mounten von Dateisystemen, transparent ftp, tar/gzip

Authentication Server

Benutzeridentität = Port zu einem Authentication Server

- vier Mengen von id-Nummern: effective user ids, effective group ids, available user ids, available group ids
- 0 identifiziert root/superuser
- effective ids: für direkte Überprüfung von Berechtigungen
- available ids: können auf Anfrage des Users in effective ids gewandelt werden

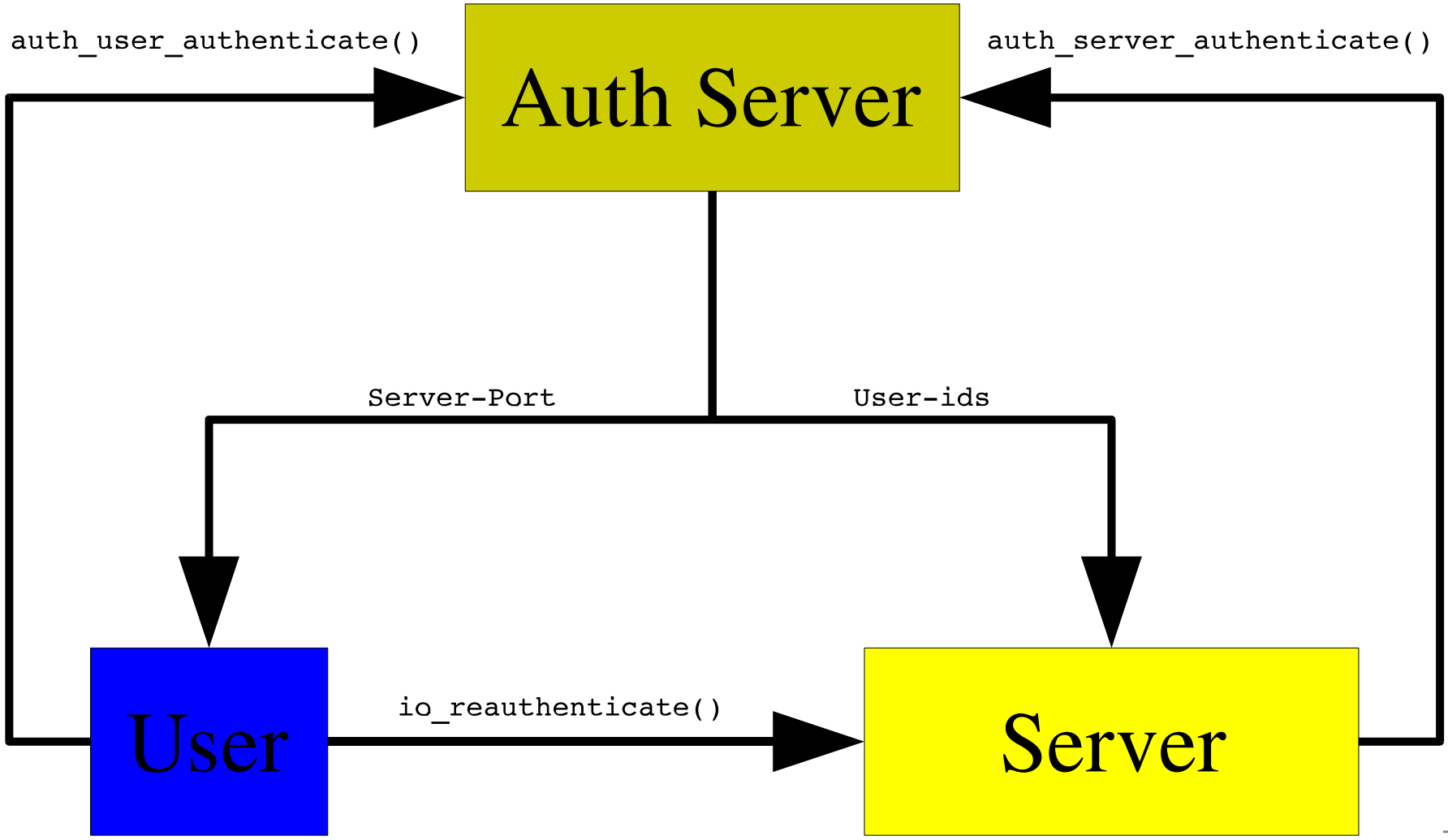
Authentication Server

Operationen mit Authentication Ports

- Mischen der ids zweier Ports in einen neuen Port
- Erzeugen eines neuen Ports mit Teilmenge der ids eines bestehenden Ports
- Erzeugen eines neuen Ports mit beliebigen ids (nur superuser)
- Erstellen einer "trusted connection" zwischen Usern und Servern

Authentication Server

Erstellen einer Trusted Connection



Process Server

- Default Process Server vergibt PID an jeden Task
- Speichert ’envp’ und ’argv’
- Erstellung eines message port (IPC)
- Nameserver für PID’s

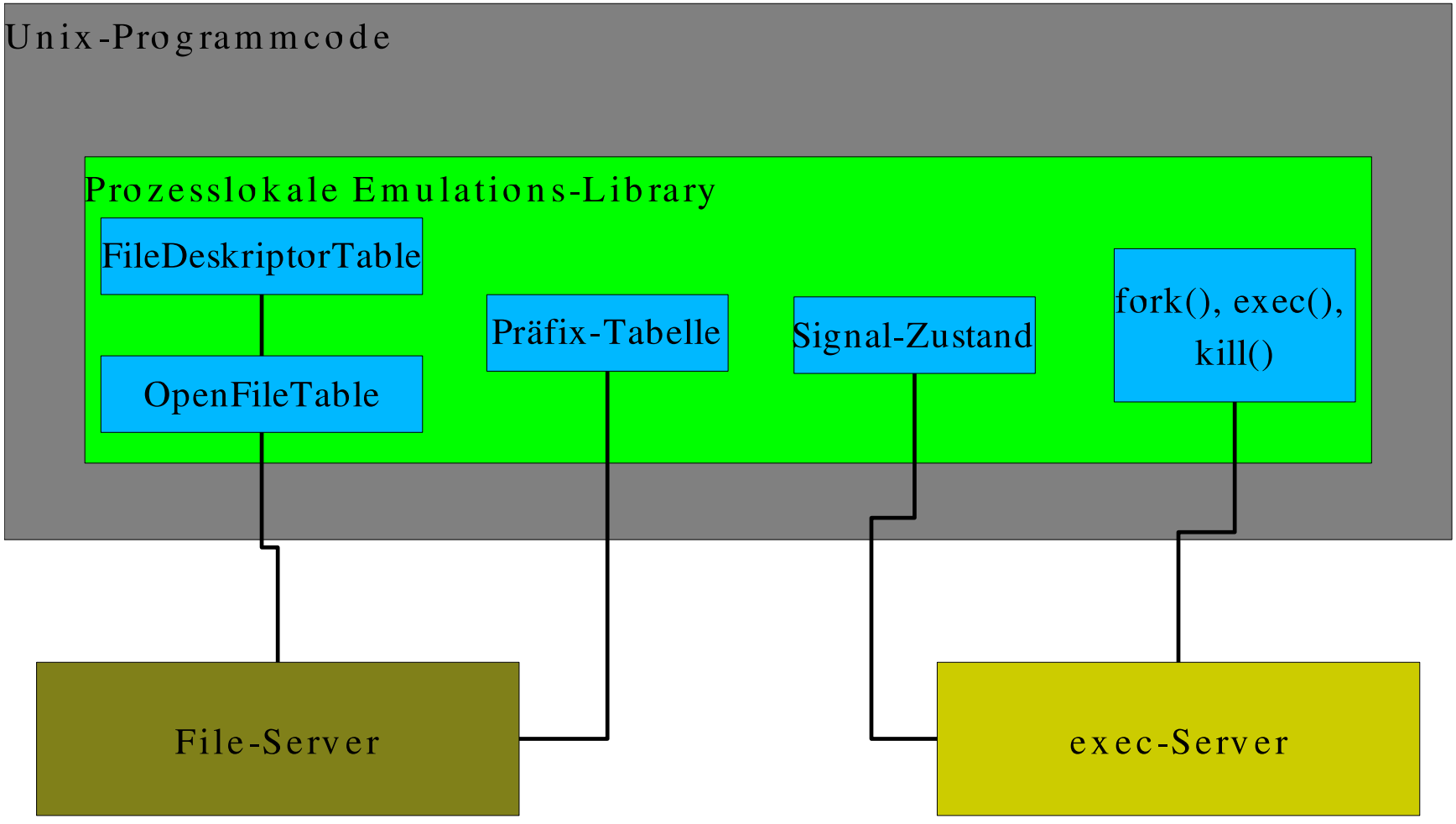
Programmausführung

execve() Systemaufruf

- Drei Komponenten beteiligt:
  - GlibC Library (argv, envp)
  - File Server (Präfix-Pfade, Berechtigungen)
  - exec Server (erstellt Task)
- exec Server unterstützt mehrere ausführbare Formate, kann '#!'-Scripte ausführen

Programmausführung

Aufbau einer Unix-Task in Hurd



Unix Look

- C Library enthält alle Funktionen von BSD und POSIX sowie einige Erweiterungen
- Mapping des POSIX Interface:
  - fd = open(name, ...): dir_lookup(.., name, .., &port)
  - read(fd, ...): io_read(port, ...)
  - write(fd, ...): io_write(port, ...)

Just say NO TO DRUGS, and maybe you won't end up like the Hurd people.

-Linus Torvalds-