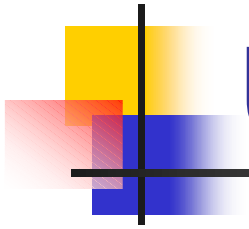
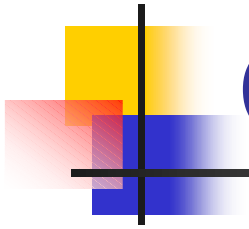
Rechteverwaltung
und
Zugriffskontrollmechanismen

Gonsu Veronique



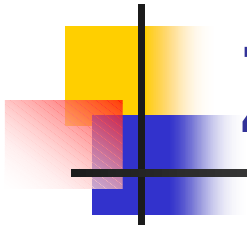
Überblick

- Zugriffskontrolle
- Acces Control List
- Problemen mit Acces Control List
- Capabilities
- Capabilities-basierte Systemen
  - EROS
  - Amoeba
- Flexible Zugriffskontrolle mit IPC Redirection

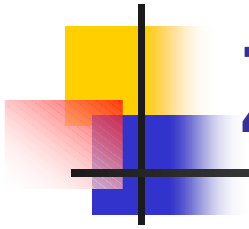


Grundbegriffe

- **Zugriff** : Art von Operationen auf Objekten
- **Objekten** : Dateien, Programmen, Netzwerk **...**
- **Eine Operation** Z.B eine Datei lesen, in einer Datei schreiben, oder auch mit anderen Programmen kommunizieren
- **Zugriffskontrollmechanismen**
  - legen fest auf welche Art die Rechte an den Objekten vergeben werden



Zugriffskontrolle



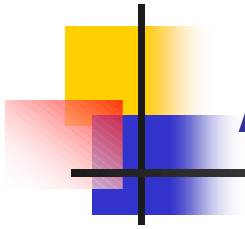
Zugriffskontrolle

- Zugriffskontrolle
  - Zugriff vehindern
    - Zugriff beschräncken
  - Zugriff gewähren

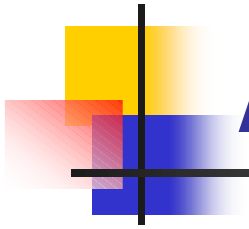
Zugriffskontrolle

- **Zugriffsmatrix**
  - **älteste und bekannteste Art der Zugriffskontrolle**
  - **Änderungen durch Eigentümer oder Administrator**
  - **Problem: können sehr groß werden**
    - **Feld für jede Subjekt-Objekt Kombination (oft leer)**

	O1	O2	O3
S1	write		
S2		read, write	
S3			read



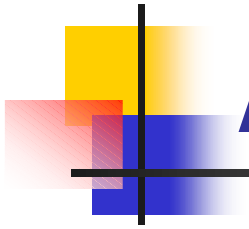
Access Control List (ACL)



Access Control List (ACL)

- Was ist ein ACL System?
  - ein System, in dem jedes Objekt mit einer Liste von Benutzern und Aktionen, die jeder Benutzer ausführen darf, verbunden ist.
  - File System von jedem Programm Zugreiffbar
  - Zugriffen verhindern und gewahren
  - aber nicht beschränken.

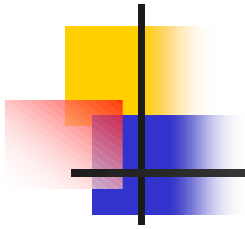
- Speicherung
  - Nur die Spalten der Zugriffmatrix mit einem Eintrag



Access Control List (ACL)

- Vorteile
  - Zu jedem Objekt sieht man leicht welche Subjekte darauf Zugriff haben
  - einfache Rechterücknahme (ACL Liste aktualisieren)

- Nachteile:
  - Länge der Liste (beim Zugriff auf einem Objekt müssen sie durchgelesen werden)
  - schwer nachzuvollziehen alle Objekte an denen ein Subjekt Zugriffsrechte hat. Da Alle ACL liste durchgelesen werden müssen

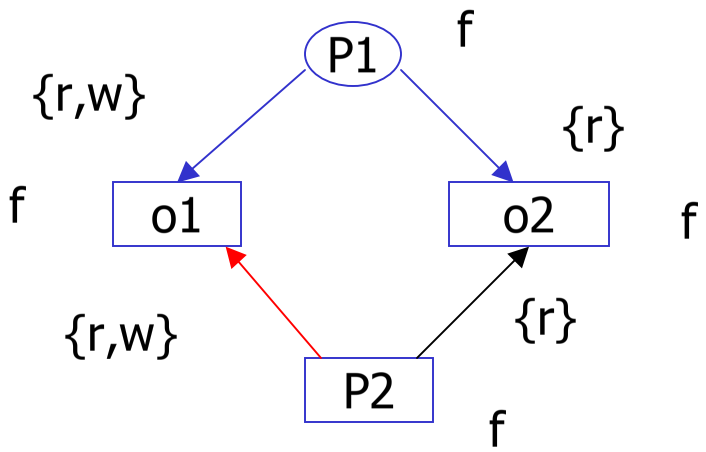


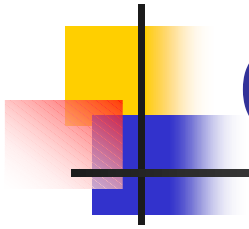
Problemen mit ACL

Problemen mit ACL

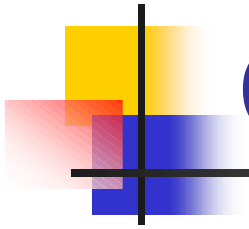
- Least Privilege
  - Alle Programmen, die unter dem Name eines Benutzers laufen, bekommen alle Rechte dieses Benutzers.
  - keine Möglichkeit zur Beschränckung

- Selective Acces right delegation





Capabilities



Capabilities

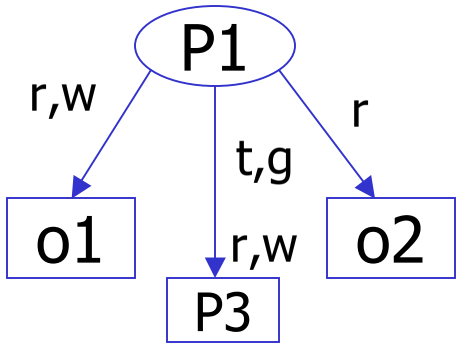
- **Ein Capability** ist ein Tickets, das den Eigentümer das Recht gibt , eine spezifische Menge von Aktionen auf ein bestimmtes Objekt duchrzuführen.

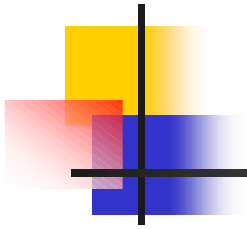
- Wie kann ein Programm Capabilities bekommen?
  - Durch sein Creator
  - Durch ein anderes Programm

- Beispiel von Capability-basierten Systemen
  - EROS : Plattform für sichere Anwendungen
  - Amoeba : Objekt-orientiertes verteiltes Betriebsystem

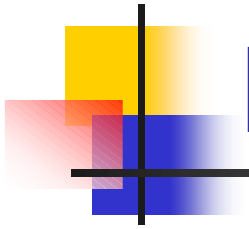
Eigenschaften von Capabilities

- Least Privilege
  - nicht mehr Authority als benötigt
- Selective Acces Right Delegation
  - Authorities delegieren zum Subprogramm



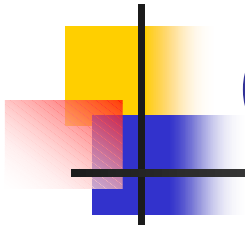


Problemen mit Capabilities

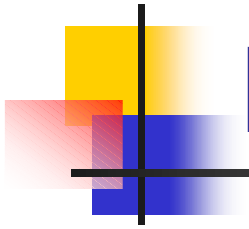


Problemen mit Capabilities

- Prinzipielles Problem:
  - Speicherung von Capabilities
  - Wierderverfügbarkeit von Capabilities nach einem Systemabsturz
- Lösung
  - In persistenten System (EROS)
  - In non-persistenten Systemen (Amoeba)



Capabilities-basierte Systemen



Persistentes System: EROS

- Persistenz bedeutet
  - der Systemzustand bleibt erhaltet (auch laufende Prozesse)
  - Selbst über einen Neustart

- Checkpointing (Mechanismus)
  - macht regelmäßig einen konsistenten Schnappschuss (snapshot) von kompletten System
  - speicher auf die Festplatte
  - während des Schnappschuss werden alle Prozesse gestoppt
  - effizient: 100ms jede 5 minuten
  - System recovery sehr schnell

Non-persitentes System: Amoeba

- Capabilities werden verschüsselt und als normale Datei gespeichert (permanent store)

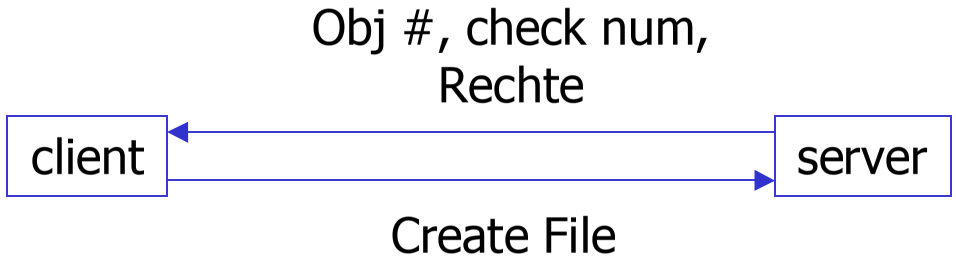
- Capabilities in Amoeba 5.2 haben folgendes Format

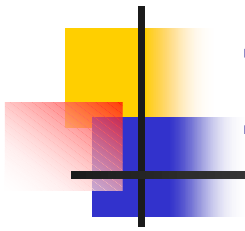
Server port	Objekt	Rechte	Check-Überprüng

  - Put-port für den Server, der das objekt verwaltet
  - Objektnummer: wichtig für den Server
  - Feld für die Rechte: 1 bit für jede erlaubte Operation
  - Zahl für die Protection

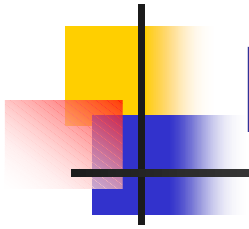
Capabilities in Amoeba

- Beispiel: neues Objekt
  - Server
    - zuffällige Zahl für Check-Feld und Objeknummer
    - speichert in Capability und in seiner Tabellen
    - alle Berechtigungsbit sind auf 1



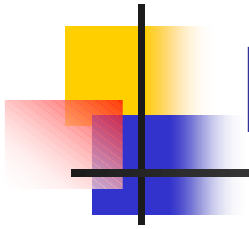


Flexible Zugriffskontrolle mit IPC Redirection



Referenzmonitor

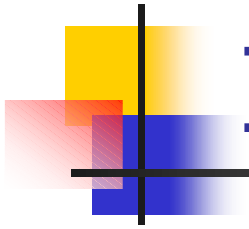
- Was ist ein Referenzmonitor ?
  - zuverlässiger Prozess
  - Aufgabe
  - entscheiden
  - Subjekt (Prozess)
  - System Policy
  - Zugriff auf ein Objekt



Referenzmonitor

- Was macht ein Referenzmonitor ?
  - Setzt die Sicherheitstrategie des Systems durch
- wie?
  - Zugriff verhindern
  - Zugriff überprüfen
  - Rechte zurücknehmen falls Änderungen in der Zugriffskontroll Policy

Flexible Zugriffskontrolle mit IPC Redirection



- IPC: inter-process communication
- In Mikro-kernel Systemen ist IPC die einzige kommunikationsmöglichkeit zwischen Prozessen, die an dem User-level implementiert wurden.
- Management von Umleitungen
  - nicht der Kernel
  - sondern User-level Prozess **Redirection Controller**.

IPC redirection Mechanism

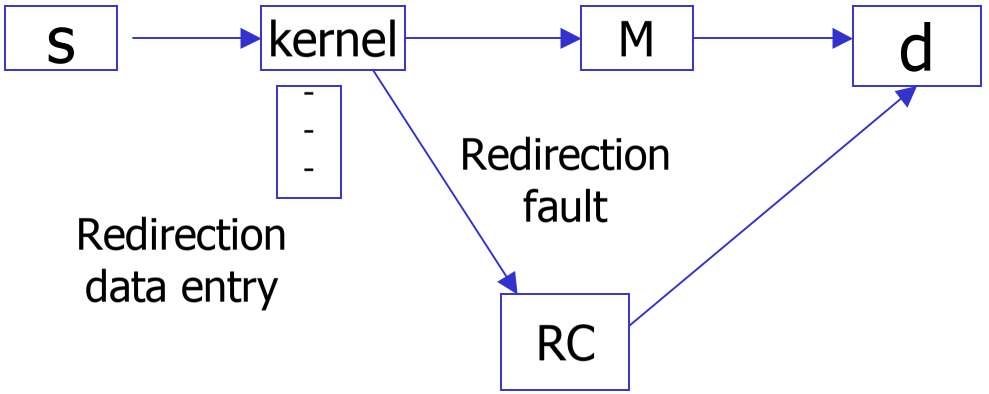
- Der Redirection Controller (RC) darf
  - Umleitunsstrategien für die Prozessen in seiner Umleitungsmenge bestimmen.
    - Umleitungsmenge
      - Menge von Prozessen
      - Redirection controller , Umleitung Policy
  - Monitoren für jeder Prozess bestimmen.
- Umleitungsfunktion R :

  $R(s,d) \rightarrow i$, mit i,s,d Prozessen

  R abbildet den Quellprozess s und den Zielprozess d mit dem Zwischenprozess i.

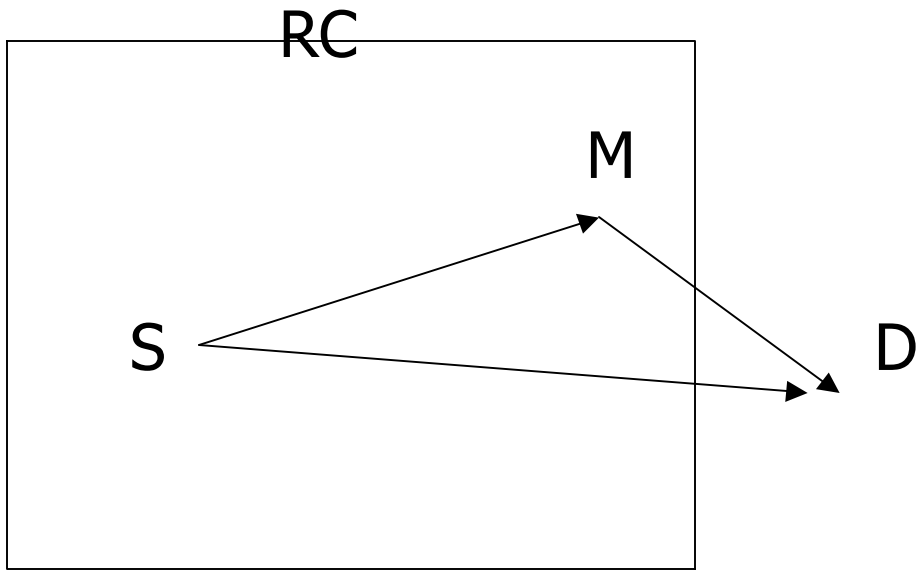
IPC Redirection mechanism

- s $\rightarrow$ d

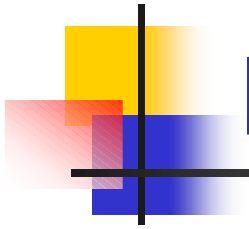


- Zwischenprozesse
  - können IPCs zwischen Quell- und Zielprozess blokieren, überprüfen, oder weiterleiten.

Beispiel: Referenzmonitor

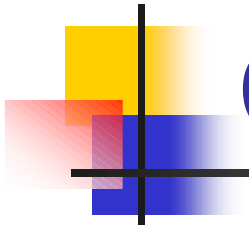


Referenzmonitor M kontrolliert Operationen von Prozessen S und D wie sie von RC spezifiziert wurden.



Beispiel: Referenzmonitor

- Prozess RC ist das redirection controller für das System.
- Prozess M ist den Monitor für Prozessen s und d.
  - R(s,_) = M. M ist den Zwischenprozess für alle IPCs mit Quellprozess s.
  - Er führt alle Zugriffskontroll für diese Prozessen aus
- Falls s ein IPC zu d senden möchte, leitet der kernel das IPC zu M um, der das IPC autorisiert und weiterleitet.
- Alle IPCs , die rückkehren, werden automatisch durch das Referenzmonitor umgeleitet.

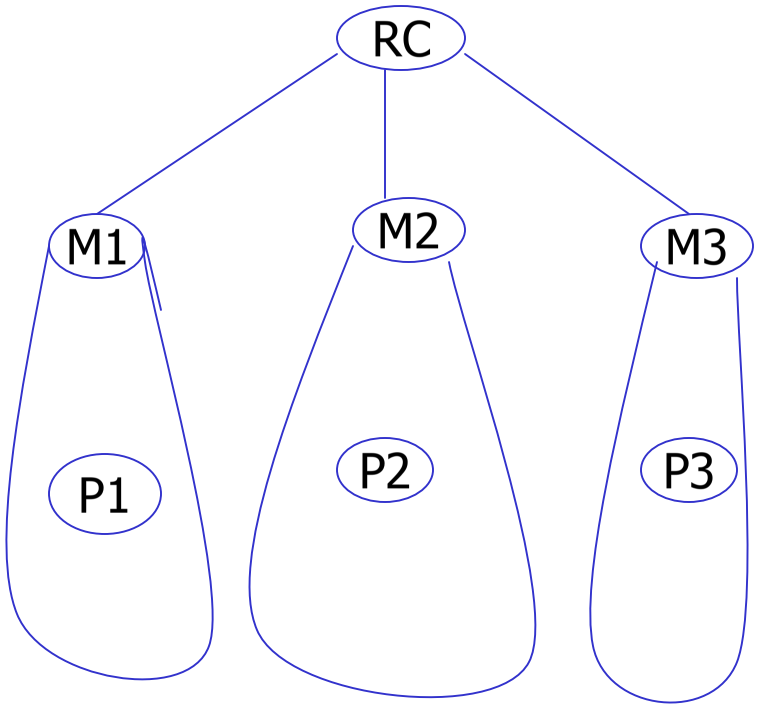


Clans & Chiefs

- Clans & Chiefs
  - volle algorithmische Kontrolle von Prozessinteraktion
  - auf einer User-definierbare aber sichere Weise
- Clan
  - Menge von Prozesse
  - geleitet von einem Chief
- Chief
  - kontrolliert alle Kommunikation mit der Außenwelt
  - Alle Nachrichten nach außerhalb des Clans werden durch Kernel zum Chief umgeleitet

Beispiel: Clans & Chiefs

- M1, M2 ,M3 chiefs
- RC chief von M1,M2,M3
- P1: R(P1,P2) = M1,
  R(P1,P3) = M1
- M1: R(M1,P1) = P1,
  R(M1,P2) = M2,
  R(M1,P3) = M3



ENDE