

8 Digitale Unterschriften (2)

- Kombination mit Verschlüsselung
 - ◆ erst signieren
 - ◆ dann mit dem öffentlichen Schlüssel des Adressaten verschlüsseln
- ▲ Reihenfolge wichtig
 - ◆ Man signiere nichts, was man nicht entschlüsseln kann
- Heute gängiges Hash-Verfahren
 - ◆ MD5
 - ◆ 128 Bit langer Hash-Wert

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.105

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

8 Digitale Unterschriften (4)

- ▲ Mögliche Probleme von Public key-Verfahren
 - ◆ Geheimhaltung des geheimen Schlüssels
(Time sharing-System, Backup; Schlüsselpaßwort / Pass phrase)
 - ◆ Vertrauen in die Programme (z.B. PGP)
 - ◆ Ausspähung während des Ver- und Entschlüsselungsvorgangs

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.107

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

8 Digitale Unterschriften (3)

- Woher weiß ich, daß ein öffentlicher Schlüssel authentisch ist?
 - ◆ Ich bekomme den Schlüssel vom Eigentümer (persönlich, telefonisch).
 - Hash-Wert auf öffentlichen Schlüsseln, die leichter zu überprüfen sind (Finger prints)
 - ◆ Ich vertraue jemandem (Bürge), der zusichert, daß der Schlüssel authentisch ist.
 - Schlüssel werden von dem Bürgen signiert.
 - Bürge kann auch eine ausgezeichnete Zertifizierungsstelle sein.
 - Netzwerk von Zusicherungen auf öffentliche Schlüssel (*Web of Trust*)
 - ◆ Möglichst weite Verbreitung von öffentlichen Schlüsseln erreichen (z.B. PGP: Webserver als Schlüsselserver)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.106

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.7 Authentisierung im Netzwerk

- Viele Klienten, die viele Dienste in Anspruch nehmen wollen
 - ◆ Dienste (*Server*) wollen wissen welcher Benutzer (*Principal*), den Dienst in Anspruch nehmen will (z.B. zum Accounting, Zugriffsschutz, etc.)
 - ◆ Im lokalen System reicht die (durch das Betriebssystem) geschützte Benutzererkennung (z.B. UNIX UID) als Ausweis
 - ◆ Im Netzwerk können Pakete abgefangen, verfälscht und gefälscht werden (einfache Übertragung einer Benutzererkennung nicht ausreichend sicher)
- Public key-Verfahren
 - ◆ Authentisierung durch digitale Unterschrift (mit geheimen Schlüssel des Senders) und Verschlüsseln (mit öffentlichem Schlüssel des Empfängers)
 - ◆ Nachteile
 - jeder Dienst benötigt sicheren Zugang zu allen öffentlichen Schlüsseln
 - Verschlüsseln und Signieren mit RSA ist sehr teuer

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.108

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1.7 Authentisierung im Netzwerk (2)

- ★ Einsatz von Authentisierungsdiensten
 - ◆ zentraler Server, der alle Benutzer kennt
 - ◆ Authentisierungsdienst garantiert einem Netzwerkdienst, daß ein Benutzer auch der ist, der er vorgibt zu sein
- Benutzerausweis
 - ◆ der Authentisierungsdienst erkennt den Benutzer anhand eines geheimen Schlüssels oder Paßworts
 - ◆ der Schlüssel ist nur dem Authentisierungsdienst und dem Benutzer bekannt
- Vorgang
 - ◆ Benutzer (*Principal*) will mit einem Programm (*Client*) einen Dienst (*Server*) in Anspruch nehmen
 - ◆ durch geeignetes Protokoll erhalten Client und Server jeweils einen nur ihnen bekannten Schlüssel, mit dem sie ihre Kommunikation verschlüsseln können (*Session key*)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.109

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Einfacher Authentisierungsdienst (2)

- ▲ Problem
 - ◆ letzte Nachricht von A an B könnte aufgefangen und später erneut ins Netz gegeben werden (*Replay attack*)
 - ◆ Folge: Kommunikation zwischen A und B kann gestört werden
- ★ Korrektur durch zusätzliches Versenden einer Verbindungsbestätigung durch B und A

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

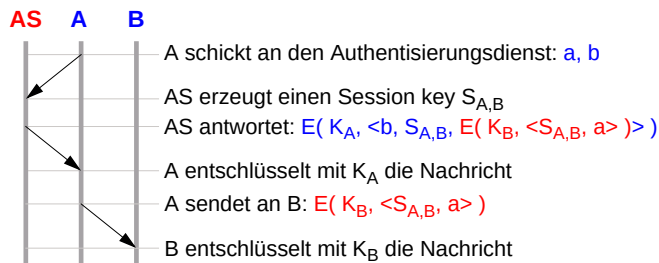
I-Security.fm 1999-02-16 13.38

I.111

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Einfacher Authentisierungsdienst

- A will den Dienst B in Anspruch nehmen (Nach Needham-Schröder):



- ◆ K_X ist der geheime Schlüssel, den nur Authentisierungsdienst und X kennen
- ◆ nach dem Protokollablauf kennen sowohl A und B den Session key
- ◆ A weiß, daß nur B den Session key kennt
- ◆ B weiß, daß nur A den Session key kennt

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

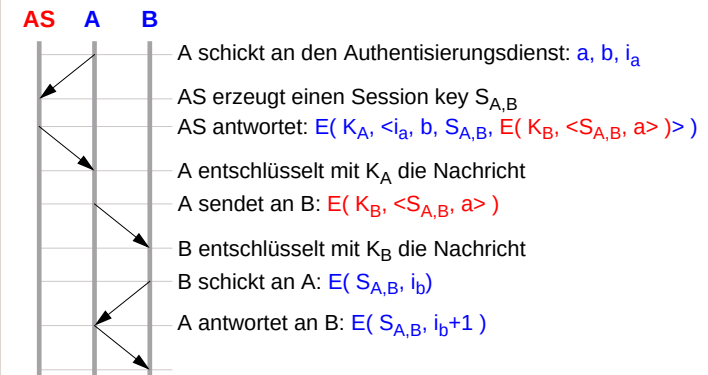
I-Security.fm 1999-02-16 13.38

I.110

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Authentisierungsdienst mit Bestätigung

- Bestätigung enthält Einmalinformation (*Nonce*)



- ◆ ein Wiedereinspielen der Nachricht $E(K_B, \langle S_{A,B}, a \rangle)$ oder $E(S_{A,B}, i_b + 1)$ wird erkannt und kann ignoriert werden

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.112

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Authentisierungsdienst mit Bestätigung (2)

▲ Problem

- ◆ Aufzeichnen von $E(K_B, \langle S_{A,B}, a \rangle)$ und
- ◆ Brechen von $S_{A,B}$ erlaubt das Aufbauen einer Verbindung.
- ◆ ein Dritter kann dann die erste Bestätigung abfangen und die zweite Bestätigung verschicken

★ Lösung

- ◆ Einführung von Zeitstempeln (*Time stamp*) und Angaben zur Lebensdauer (*Expiration time*)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

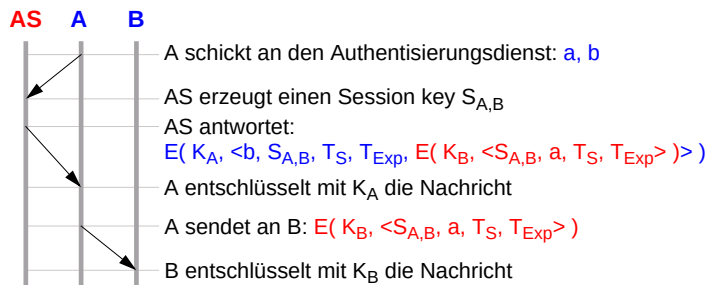
I-Security.fm 1999-02-16 13.38

I.113

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Authentisierungsdienst mit Zeitstempeln

■ Authentisierungsdienst versieht seine Nachrichten mit Zeitstempeln



- ◆ T_S = Zeitstempel der Nachrichtenerzeugung
- ◆ T_{Exp} = maximale Lebensdauer der Nachricht
- ◆ aufgezeichnete Nachricht kann nach kurzer Zeit (z.B. 5min) nicht noch einmal zum Aufbau einer Verbindung verwendet werden

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.114

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Beispiel: Kerberos

■ Kerberos V5

- ◆ Softwaresystem implementiert Weiterentwicklung des Needham-Schröder-Protokolls
- ◆ entwickelt am MIT seit 1986

■ Ziel

- ◆ Authentisierung und Erzeugung eines gemeinsamen Schlüssels durch den vertrauenswürdigen Kerberos-Server

■ Idee

- ◆ Trennung von Authentisierungsdienst und Schlüsselerzeugung
- ◆ reduziert die nötige Übertragung einer Identifikation oder eines Paßworts zum Kerberos-Server

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

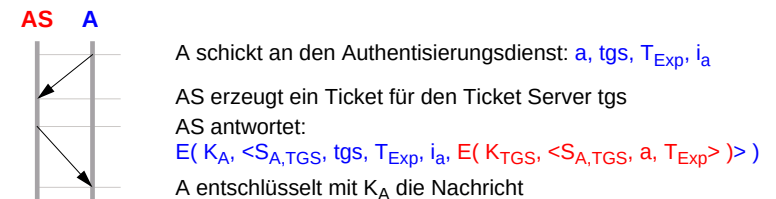
I-Security.fm 1999-02-16 13.38

I.115

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Beispiel: Kerberos (2)

■ Benutzer holt sich zunächst ein Ticket vom Authentisierungsdienst



- ◆ das Ticket besteht aus $\langle S_{A,TGS}, a, T_{Exp} \rangle$
- ◆ es enthält einen Session key für die Kommunikation mit einem Ticket granting server, der dann die Verbindung zu einem Netzwerkdienst bereitstellen kann

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

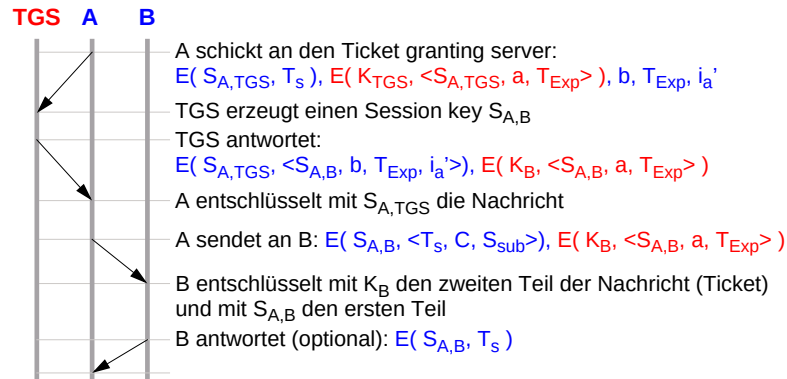
I-Security.fm 1999-02-16 13.38

I.116

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Beispiel: Kerberos (3)

- Authentisierungsdienst versieht seine Nachrichten mit Zeitstempeln



- ◆ C = Checksumme zur Überprüfung der richtigen Entschlüsselung
- ◆ A kann mehrere Verbindungen mit seinem Ticket öffnen

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

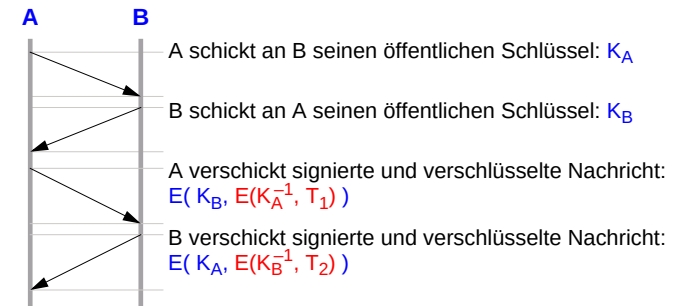
I-Security.fm 1999-02-16 13.38

I.117

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Austausch öffentlicher Schlüssel

- A und B tauschen ihre öffentlichen Schlüssel aus



- ▲ Problem

- ◆ A und B können nicht sicher sein, daß der öffentliche Schlüssel wirklich vom jeweils anderen stammt

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.119

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Beispiel: Kerberos (4)

- Unterscheidung zwischen Benutzer (User) und Benutzerprogramm (Client)

- ◆ Wie kann ein Benutzerprogramm seinen Benutzer identifizieren?

- ◆ Geheimer Schlüssel vom Benutzer (K_X) hängt von einem Paßwort ab
- ◆ mittels einer Einwegfunktion wird aus dem Paßwort der Schlüssel K_X erzeugt
- ◆ Benutzerprogramm braucht also das Paßwort zur Verbindungsaufnahme

- Beispiel: *kinit*, *klogin*

- ◆ Anmeldung beim Authentisierungsdienst mit *kinit* und Paßworteingabe
- ◆ Ticket wird im Benutzerkatalog gespeichert
- ◆ *klogin* erlaubt das Einloggen auf einem entfernten Rechner mit Datenverschlüsselung und ohne Paßwort

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

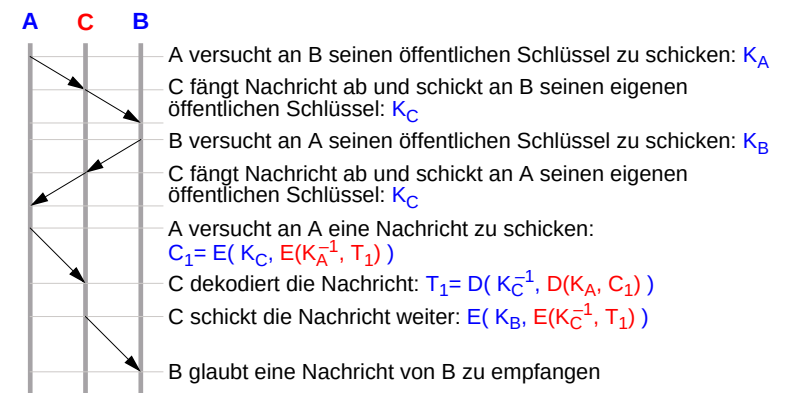
I-Security.fm 1999-02-16 13.38

I.118

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Austausch öffentlicher Schlüssel (2)

- Aktiver Mithörer C fängt Datenverbindungen ab (Man in the middle attack)



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

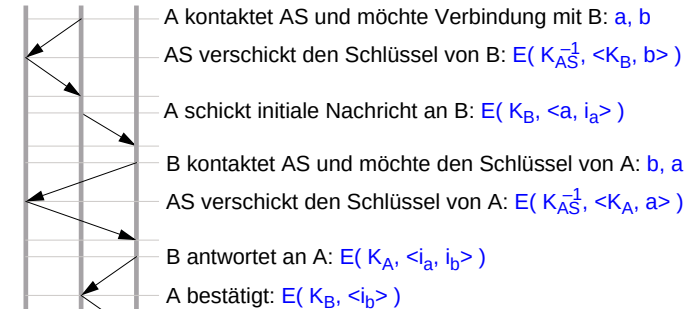
I.120

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Austausch öffentlicher Schlüssel (3)

■ Einsatz eines Authentisierungsdienstes

AS A B



▲ Replay-Probleme

- ◆ Hinzunahme von Zeitstempel und Lebendauer

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

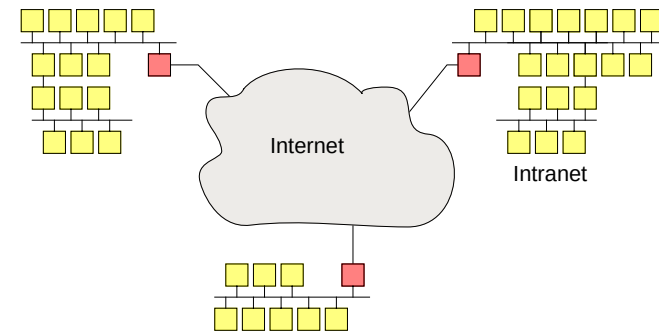
I.121

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.8 Firewall (2)

■ Virtual private network

- ◆ Verbinden von Intranet-Inseln durch spezielle Tunnels zwischen Firewalls
- ◆ getunnelter Datenverkehr wird verschlüsselt
- ◆ Benutzer sieht ein „großes Intranet“ (nur virtuell vorhanden)



SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.123

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.8 Firewall

■ Trennung von vertrauenswürdigen und nicht vertrauenswürdigen Netzwerksegmenten durch spezielle Hardware (*Firewall*)

- ◆ Beispiel: Trennen des firmeninternen Netzwerks (Intranet) vom allgemeinen Internet

■ Funktionalität

- ◆ Einschränkung von Diensten
 - von innen nach außen, z.B. nur Standarddienste
 - von außen nach innen, z.B. kein Telnet, nur WWW
- ◆ Paketfilter
 - Filtern „defekter“ Pakete, z.B. SYN-Pakete
- ◆ Inhaltsfilter
 - Filtern von Pornomaterial aus dem WWW oder News
- ◆ Authentisieren von Benutzern vor der Nutzung von Diensten

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.122

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

I.9 Richtlinien für den Benutzer

1 Paßwörter

■ Wahl eines Paßworts

- ◆ hinreichend komplexe Paßwörter wählen
- ◆ Schutz vor Wörterbuchangriffen
- ◆ verschiedene Paßwörter für verschiedene Aufgaben (z.B. PPP Paßwort ungleich Benutzerpaßwort)

■ Aufbewahrung

- ◆ möglichst nirgends aufschreiben
- ◆ nicht weitergeben
- ◆ kein Abspeichern auf einem Windows-Rechner (Option immer wegklicken)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.124

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Paßwörter

- Eingabe
 - ◆ niemals über eine unsichere Rechnerverbindung eingeben
 - **ftp, telnet, rlogin** Dienste vermeiden
 - nur sichere Dienste verwenden: **ssh, slogin**
 - Datenweg beachten, über den das Paßwort läuft: ein unsicheres Netzwerk ist bereits genug
- Änderung
 - ◆ Paßwörter regelmäßig wechseln
 - ◆ alte Paßwörter nicht wiederverwenden

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.125

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Schlüsselhandhabung

- Einsatz von PGP oder S/MIME
 - ◆ Zugang zu den privaten Schlüsseln für andere verhindern
 - ◆ Dateirechte auf der Schlüsseldatei prüfen
 - ◆ privater Schlüssel nur auf Diskette
 - ◆ Passphrase wie ein Paßwort behandeln
 - ◆ privaten Schlüssel nie über unsichere Netze transportieren

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.126

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 E-Mail

- Authentisierung
 - ◆ Bei elektronischer Post ist der Absender nicht authentisierbar
 - ◆ Digitale Unterschriften einsetzen (z.B. mit PGP oder S/MIME)
- Abhören
 - ◆ Elektronische Post durchläuft viele Zwischenstationen und kann dort jeweils gelesen und verfälscht werden
 - ◆ Verschlüsselung einsetzen (z.B. mit PGP oder S/MIME)

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.127

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Programmierung

- S-Bit Programme vermeiden
 - ◆ Oft kann das S-Bit durch geschickte Vergabe von Benutzergruppen an Dateien vermieden werden
- Verwendung zusätzlicher Rechte (z.B. durch S-Bit) nur in Abschnitten
 - ◆ Trusted Computing Base (TCB)

```
seteuid(getuid());/* am Programmanfang Rechte wegnehmen */
...
seteuid(0);      /* setzt root Rechte */
fd = open("/etc/passwd", O_RDWR);
seteuid(getuid());/* nimmt root Rechte wieder weg */
...
```
- Sorgfältige Programmierung
 - ◆ Funktionen wie **strcpy, strcat, gets, sprintf, scanf, sscanf, system, popen** vermeiden oder durch **strncpy, fgets, snprintf** ersetzen

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13.38

I.128

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 World Wide Web

- Cookies
 - ◆ Akzeptieren von Cookies erlaubt einer Web site die angesprochenen Seiten genau einem Benutzer zuzuordnen
 - ◆ funktioniert über Sessions hinweg
- JavaScript
 - ◆ schwere Sicherheitslücken erlauben es, alle für den Benutzer lesbare Dateien an einen dritten weiterzugeben
 - ◆ Ausschalten!
- Abhören
 - ◆ WWW Verbindungen können abgehört werden
 - ◆ keine privaten Daten, wie z.B. Kreditkartennummern übertragen
 - ◆ HTTPS mit SSL-Verschlüsselung benutzen; https URLs

SP I

Systemprogrammierung I

© Franz J. Hauck, Universität Erlangen-Nürnberg, IMMD IV, 1998 / 1999

I-Security.fm 1999-02-16 13:38

I.129

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

J.1 Motivation

- Entwicklung des UNIX-Systemkerns
 - ◆ heute nicht mehr so einfach aufgebaut und leicht modifizierbar wie früher
 - ◆ früher auf PDP11 mit 32k HSP lauffähig, heute 1,5–2 MB Speicher (inkl. Puffer etc.) notwendig
 - ◆ viele neue Funktionen werden kritiklos in den Kern eingebaut, weil dort die notwendigen Information leicht zugänglich sind
 - ◆ Aufblähung des Kerns, Abstraktionen und Strukturierung werden mehr und mehr zerstört
 - ◆ für Einsatz von UNIX auf Multiprozessorsystemen sind aufwendige Modifikationen notwendig
 - Kern-Monitor muß in Teile zerlegt werden oder ganz aufgegeben werden
 - an Stellen, an denen die Monitor-Eigenschaft aufgegeben wird, ist aufwendige Koordinierung erforderlich

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.2

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

J Mach

- Fallbeispiel: *Mach* Betriebssystem
 - ◆ Mikrokern
 - ◆ Unterstützung für Multiprozessoren
 - ◆ neues, hardwareunabhängiges Konzept der virtuellen Speicherverwaltung
 - ◆ *Capability*-basierte Interprozeßkommunikation – transparent erweiterbar für Netzwerk-Kommunikation
 - ◆ Modularer Aufbau, einfache Erweiterbarkeit (nur die wichtigsten Funktionen sind im Mach-Kern realisiert)
 - ◆ Betriebssystemumgebung wird durch eine Reihe von *Servern* realisiert
- Geschichte
 - ◆ Entwicklung 1985–1994 an der Carnegie Mellon Univ. (CMU)
 - ◆ Basis für OSF/1 (Open Software Foundation), NeXT-OS (Next/Apple)
 - ◆ Bedeutung heute eher gering

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

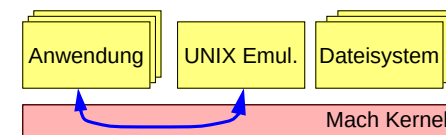
J-Mach.fm 1999-02-09 15:39

J.1

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

J.1 Motivation (2)

- Architektur von Mach
 - ◆ Mikrokern
 - ◆ Betriebssystemumgebung wird durch eine Reihe von *Servern* realisiert, die über die Basis-Mechanismen des Mach-Kerns angesprochen werden können



- ◆ Beispiele:
 - Dateisystem
 - Netzwerk-Kommunikation (z.B. TCP/IP)
 - Scheduling
 - UNIX-Emulation

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.3

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

J.2 Abstraktionen des Mach-Kerns

- **Task:** Ablaufumgebung für *Threads*
 - ◆ virtueller Adreßraum
 - ◆ gesicherter Zugriff zu Systemressourcen (Prozessoren, *Port-Capabilities*, Speicher)
- **Thread:** Aktivitätsträger innerhalb einer *Task*
 - ◆ beschreibbar durch
 - einen unabhängigen Programmzähler
 - eigenen Stack-Bereich innerhalb des virt. Adreßraums der *Task*
 - ◆ alle *Threads* einer *Task* haben gemeinsamen Zugriff zu allen *Task*-Ressourcen
- **Port:** Kommunikationskanal mit Warteschlange für Nachrichten
 - ◆ wird vom MACH-Kern verwaltet

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15.39

J.4

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

J.3 Tasks und Threads

1 Task

- Betriebsumgebung (System-Ressourcen) für Aktivitätsträger
 - ◆ virtueller Adreßraum
 - ◆ Zugriffsrechte
 - ◆ Betriebsmittel-Informationen
 - ◆ kein Programmablauf und keine Register

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15.39

J.6

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

J.2 Abstraktionen des Mach-Kerns (2)

- **Message:** Menge von Datenobjekten für die Kommunikation zwischen *Threads*
 - ◆ Messages können typisiert werden
 - ◆ können max. den gesamten virt. Adreßraum umfassen
 - ◆ können Zeiger und *Capabilities* für *Ports* enthalten
- Operationen
 - ◆ Operationen auf einem Objekt (ausgenommen *Messages*) werden durch Versenden von *Messages* an *Ports* ausgeführt, die dieses Objekt repräsentieren.
 - ◆ Beispiel:
 - Beim Generieren einer *Task* erhält man die Zugriffsrechte auf einen *Port* dieser *Task*.
 - Durch *Messages* an diesen *Port* kann die *Task* beeinflusst werden.

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15.39

J.5

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Thread

- Aktivitätsträger und seine Ablaufumgebung
 - ◆ Registersatz
 - ◆ Stack
 - ◆ Programmzähler
- Innerhalb einer *Task* können beliebig viele *Threads* existieren
 - ◆ In einer echten Multiprozessorumgebung können verschiedene *Threads* einer *Task* parallel auf mehreren Prozessoren ablaufen
 - ◆ Ein herkömmlicher UNIX-Prozeß entspricht einer MACH-*Task* mit einem *Thread*

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15.39

J.7

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interprocess Communication – IPC

- nachrichtenorientierte Kommunikation
- Port
 - ◆ geschütztes Objekt des MACH-Kerns
 - ◆ kann Nachrichten (*Messages*) von einem Sender entgegennehmen und an einen Empfänger ausliefern
 - ◆ zu einem *Port* können beliebig viele Sender, aber nur ein Empfänger existieren
 - ◆ Zugriff auf einen *Port* erhält man durch Empfang einer Mitteilung, die ein *Port-Zugriffsrecht (Capability)* enthält

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.8

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interprocess Communication – IPC (3)

- Beispiel
 - ◆ Beim Erzeugen eines Threads erhält man Senderechte auf die *Ports* dieses Threads. Durch Mitteilungen an diese *Ports* kann der Thread manipuliert werden (suspendieren, deblockieren, vernichten, ...)
- Ablauf einer typischen Interaktion
 - ◆ Ein Thread von Task A schickt eine Nachricht an einen Port
 - ◆ Task B ist der Empfänger für diesen Port

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.10

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interprocess Communication – IPC (2)

- Message
 - ◆ Ansammlung von Datenobjekten
 - ◆ unstrukturiert oder strukturiert (durch einen Typ näher gekennzeichnet), z.B.:
 - Daten mit Typinformation
 - Zeiger auf Adreßraumbereiche
 - *Port-Zugriffsrechte (Capabilities)*
- IPC ist in MACH der allgemeine Mechanismus zur Ausführung von Operationen auf Objekten
 - ◆ Objekten (z. B. *Tasks* und *Threads*) sind Ports zugeordnet
 - ◆ Operationen auf Objekten werden durch Nachrichten an diese Ports ausgeführt
 - ◆ zuverlässige Nachrichtenübertragung

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

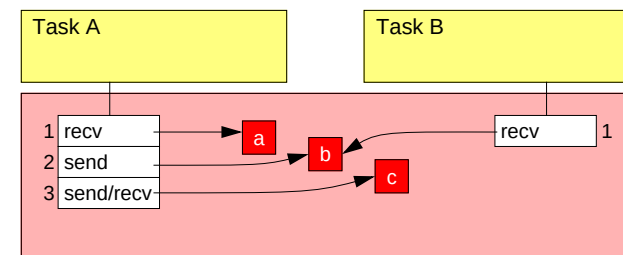
J-Mach.fm 1999-02-09 15:39

J.9

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interprocess Communication – IPC (4)

- Capabilities werden ähnlich wie Dateideskriptoren verwaltet



- ◆ Jeder Task hat einen privaten Deskriptor für eine Portcapability
- ◆ Deskriptor ist mit Rechten verbunden: Senderecht (send), Empfangsrecht (recv)

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

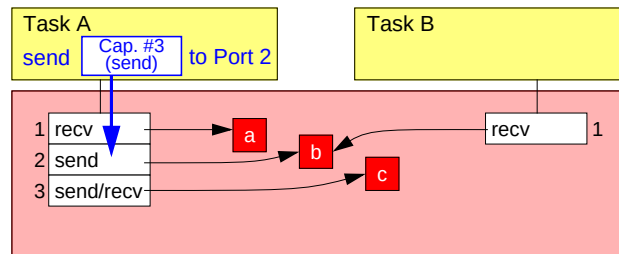
J-Mach.fm 1999-02-09 15:39

J.11

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interprocess Communication – IPC (5)

- Task A erzeugt eine Nachricht und sendet sie



- ◆ Adressat ist Port mit Deskriptor 2, Port b

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

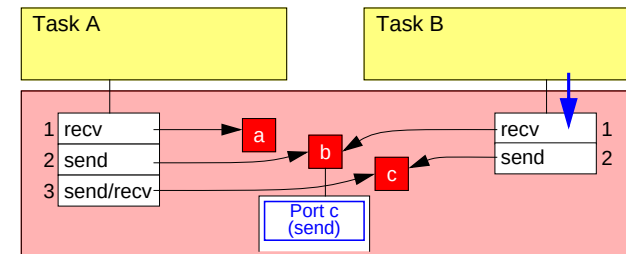
J-Mach.fm 1999-02-09 15:39

J.12

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interprocess Communication – IPC (7)

- Task B möchte eine Nachricht empfangen



- ◆ Task B benötigt neuen Deskriptor für die Capability in der Nachricht
- ◆ Deskriptor 2 mit dem entsprechenden Recht wird erzeugt

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

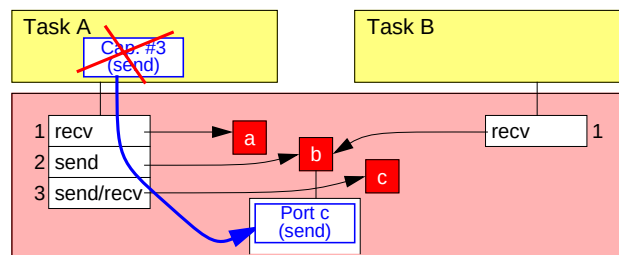
J-Mach.fm 1999-02-09 15:39

J.14

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interprocess Communication – IPC (6)

- Einstellen der Nachricht in der Port-Warteschlange



- ◆ Nachricht wird in die Warteschlange des Port b eingefügt
- ◆ übermittelte Capability wird in Portadresse umgesetzt
- ◆ Nachricht steht jetzt zum Empfang bereit
- ◆ Task A kann den Nachrichtenspeicher löschen oder wiederverwenden

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

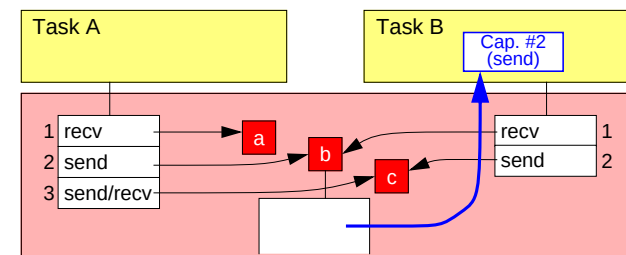
J-Mach.fm 1999-02-09 15:39

J.13

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interprocess Communication – IPC (8)

- Task B empfängt Nachricht



- ◆ Task B erhält die Nachricht
- ◆ Portadresse wird in Deskriptor übersetzt
- ◆ Nachricht wird aus der Warteschlange entfernt
- ◆ Task B hat nun Zugriff auf den Port c

SP I

Systemprogrammierung I
© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.15

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Übertragen einer „großen“ Nachricht

■ Ablauf

- ◆ Task A sendet Message an *Port P1*
- ◆ der Datenbereich der *Message* wird temporär in den Adreßraum des Mach-Kerns abgebildet (= Senden an *Port*)
- ◆ im Adreßraum der Task A wird der Datenbereich als "*copy on write*" markiert (d. h. für Seiten, auf die nachfolgend schreibend durch A zugegriffen wird, wird eine Kopie erzeugt und diese wird in den virtuellen Adreßraum von A übernommen)
- ◆ Task B empfängt die Mitteilung, indem der Datenbereich in den virt. Adreßraum von B abgebildet wird
- ◆ Aus dem Adreßraum des Mach-Kerns wird der Datenbereich wieder entfernt
- ◆ Auch bei *Task B* ist der Datenbereich als "*copy on write*" markiert

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.16

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Motivation

- heute viele unterschiedliche MMUs verbreitet
- weitgehend äquivalente Funktionalität
 - ◆ unterschiedliche virtuelle Adreßräume, beschrieben durch Datenstrukturen (z. B. *Page table*)
 - ◆ pro Seite:
 - Zeiger auf physikalische Adresse der Seite
 - *Valid bit* zeigt an, ob die physikalische Seite gültig ist
 - *Protection bits* (z. B. *read/write, user/kernel*)
 - *Dirty bit* (Seite ist modifiziert)
 - *Reference bit* (Seite wurde angesprochen)

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.18

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

J.4 Virtuelle Speicherverwaltung (VM)

■ Eigenschaften

- ◆ portabel
- ◆ hardwareunabhängig
- ◆ unterstützt Multiprozessoren: parallel ablauffähig implementiert
- ◆ unabhängige *Pager*

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.17

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Motivation (2)

- Zwei mögliche Vorgehensweisen zur Unterstützung von virtuellem Speicher
 - ◆ Mechanismen, die möglichst gut den Eigenschaften der verwendeten MMU entsprechen
 - System nicht portabel (z. B. VMS für VAX)
 - effizient
 - ◆ Beschränkung auf Eigenschaften, die einfach auf jeder Hardware realisierbar sind
 - System weniger leistungsfähig und ineffizient (z. B. UNIX in 4.3bsd)
 - aber portabler

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.19

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Realisierung

- Ziel
 - ◆ möglichst flexible Unterstützung von virtuellem Speicher auf möglichst vielen Architekturen
- Vorgehensweise
 - ◆ Unterteilung der Implementierung
 - architekturabhängiger Teil
 - architekturunabhängiger Teil
 - externe Pager

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15.39

J.20

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Realisierung (3)

- Architekturunabhängiger Teil
 - ◆ realisiert Benutzerschnittstelle und Funktionsumfang
 - ◆ Verwaltung der architekturunabhängigen Datenstrukturen
 - ◆ alle notwendigen Abbildungsinformationen für jede Seite des Systems können aus den architekturunabhängigen Datenstrukturen gewonnen werden
- externe Pager
 - ◆ Seitenein- und -auslagerung wird durch die virtuelle Speicherverwaltung (VM) von sogenannten *Pagern* über Messages angefordert
 - entscheiden nicht über Ein- und Auslagerungsstrategie
 - entscheiden wohin ausgelagerte Seiten transportiert und woher eingelagerte Seiten bezogen werden (z.B. Platte, Netzwerk etc.)

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15.39

J.22

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Realisierung (2)

- Architekturabhängiger Teil
 - ◆ Verwaltung der von der Hardware benötigten Adreßabbildungstabellen (*Physical address maps, pmaps*)
 - entsprechen den MMU-Tabellen (z. B. *VAX page table* oder eine Menge von Segmentregistern beim IBM PC/RT)
 - ◆ Abbildung von Mach-Seiten in eine oder mehrere Kacheln
 - ◆ konstruiert nur einen Teil der Adreßabbildung
 - ◆ Zugriff auf weitere Adressen kann bei *Page fault* ermöglicht werden
 - ◆ kleine Tabellen trotz großem (dünn besetztem) Adreßraum

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15.39

J.21

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Realisierung (4)

- Lazy evaluation
 - ◆ Speicher (Haupt- und Hintergrundspeicher!) wird erst belegt, wenn eine Seite benutzt wird
 - ◆ *Map-on-Reference*: Page tables werden nur für aktuell benötigten Speicher bereitgestellt
 - ◆ *Copy-on-write*: Seiten werden solange gemeinsam benutzt wie möglich; bei modifizierendem Zugriff werden Nutzer voneinander getrennt

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15.39

J.23

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Datenstrukturen

■ Memory Object

- ◆ repräsentiert einen Speicherbereich, der ganz oder teilweise in einen Adreßraum eingeblendet werden kann
- ◆ Bindeglied zwischen virtuellem Adreßraum und Hintergrundspeicher
- ◆ das *Memory object* kennt oder implementiert einen Pager, der weiß woher die Seiten des Speicherbereichs beschafft werden können und was im Falle eines *Pagout* zu tun ist
- ◆ Pager (intern oder extern) realisiert Ein-/Auslagerung von Seiten
- ◆ *Memory object* hält Liste von Seiten, die momentan im Hauptspeicher vorhanden sind (*cached pages*)

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.24

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Datenstrukturen (3)

■ Sharing Maps

- ◆ zur Realisierung von *Shared memory*
- ◆ Zwischenstufe zwischen *Address map* und *Memory object*
- ◆ Änderungen in *Address map*-Eintrag eines Objekts müssen nicht in den *Address maps* mehrerer Tasks nachgetragen werden

■ Shadow Object

- ◆ wird wie ein *Memory object* benutzt
- ◆ zur Realisierung von *Copy-on-write*
- ◆ nimmt Seiten auf, die kopiert werden mußten
- ◆ verweist für unveränderte Seiten auf das Originalobjekt

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.26

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Datenstrukturen (2)

■ Address Map

- ◆ repräsentiert einen Adreßraum
- ◆ als verkettete Liste von *Address map entries* realisiert
- ◆ jeder Eintrag bildet einen (virtuellen) Speicherbereich mit gleichen Eigenschaften (Schutz, Vererbung) in einen Abschnitt eines *Memory objects* ab
- ◆ *Address map entry* enthält:
 - Vorwärts- und Rückwärts-Verzeigerung
 - Start- und Endadresse des Speicherbereichs im virtuellen Adreßraum
 - aktuelle und maximale Zugriffsrechte
 - Regel für Vererbung des Speicherbereichs an neue Task
 - Zeiger auf *Memory object*
 - Offset dieses Speicherbereichs im entsprechenden *Memory object*

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.25

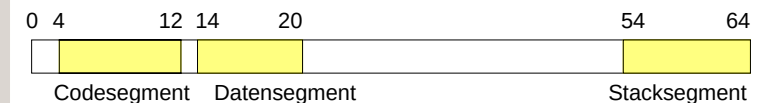
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Datenstrukturen (4)

■ Resident Page Table

- ◆ Verwaltung des Hauptspeicher-Caches für *Memory object*-Seiten
- ◆ Verkettung in Listen

■ Beispiel eines Adreßraums (z.B. einer Task)



- ◆ Abbildung des Codesegments in eine Programmdatei
- ◆ Abbildung des Daten- und Stacksegments in einen *Swap space*

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

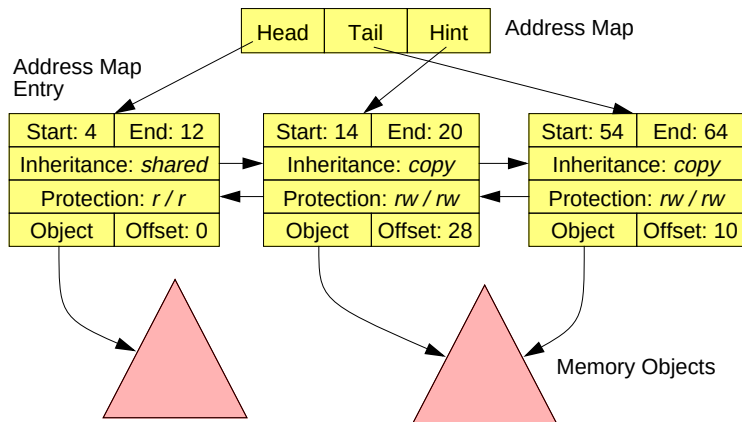
J-Mach.fm 1999-02-09 15:39

J.27

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Datenstrukturen (5)

■ Datenstrukturen der Adreßraumabbildung: *Address map*



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.28

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Benutzerschnittstelle

■ MACH-VM stellt folgende Funktionen zur Verfügung:

- ◆ virtuelle Speicherbereiche belegen und freigeben
- ◆ Schutzparameter für einzelne virtuelle Speicherbereiche individuell einstellen (innerhalb vorgegebener Schranken: *current / maximum protection*)
- ◆ Vererbung an neue Task einzeln einstellbar
 - keine Vererbung
 - Copy-on-write (default)
 - Shared
- ◆ Zugriffsmöglichkeit auf Speicherabbildung anderer Tasks (Debugger!)
- ◆ Status- und Statistikabfragen

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

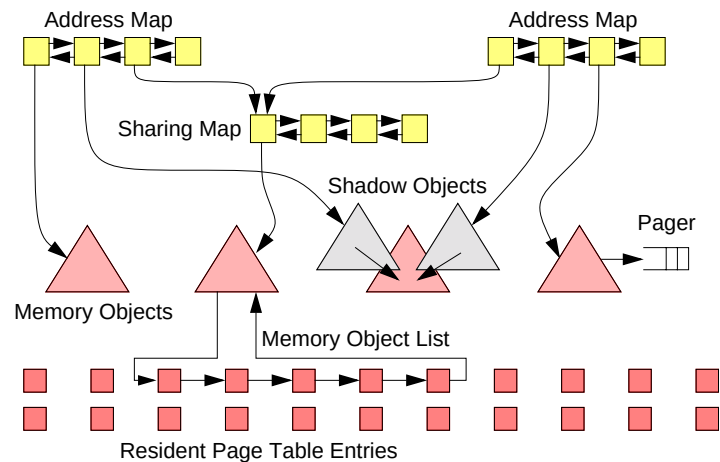
J-Mach.fm 1999-02-09 15:39

J.30

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Datenstrukturen (6)

■ Beispielhafter Gesamtüberblick



SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.29

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Benutzerschnittstelle (2)

■ Anwendungsbeispiel

- ◆ *Memory mapped files* für Benutzer und Kern
- ◆ Realisierung: Memory object mit entsprechendem Pager
- ◆ Kern-Anwendungen:
 - Programme laden
 - *Shared libraries*
 - Dateisystem
- ◆ Benutzeranwendungen:
 - neue *Stdio* mit Dateizugriff über *Mapped file*
 - Dateizugriff über Pointer

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.31

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Benutzerschnittstelle (3)

- Erfahrungen mit verschiedenen Hardware-Architekturen
 - ◆ Virtuelle Speicherverwaltung von Mach ist
 - portabel
 - stellt wenig Anforderungen an die Hardware-Plattform
 - und wurde auf einer Reihe von Architekturen implementiert
 - ◆ gute Basis für Untersuchungen von VM-Problemen bei verschiedenen Architekturen

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.32

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 VM auf Multiprozessoren (2)

- ◆ viele heutige Multiprozessorarchitekturen ignorieren dieses Problem!
- Intuitive Lösungsidee
 - ◆ nach einer Änderung der VM-Abbildung werden durch einen Befehl die TLBs der anderen Prozessoren invalidiert
- ▲ Problem
 - ◆ die meisten MPS-Systeme erlauben es einem Prozessor nur, seinen eigenen TLB, aber nicht die anderer Prozessoren zu invalidieren
- weitergehende Lösungsideen
 - ◆ Hardware-Unterstützung implementieren oder
 - ◆ durch Software-Interrupts die anderen Prozessoren zum Invalidieren ihrer TLBs bewegen

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.34

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 VM auf Multiprozessoren

- ★ Problem: *Translation Lookaside Buffer (TLB)* -Inkonsistenz
 - ◆ TLB = Cache für die Adreßdaten der SKT, um eine schnelle Adreßübersetzung ohne zusätzliche Hauptspeicher-Zugriffe durch die MMU zu ermöglichen
 - ◆ bei *Shared memory*-Multiprozessoren werden normalerweise Strategien implementiert, um die Caches der Prozessoren konsistent zu halten
 - ◆ analog ist es notwendig die TLBs der Prozessoren konsistent zu halten, wenn Threads einer Task parallel auf verschiedenen Prozessoren ablaufen
 - wenn sich die architekturabhängigen VM-Datenstrukturen einer Task ändern, müssen die TLBs aller Prozessoren aktualisiert werden, auf denen Threads der Task ablaufen
 - in anderen Betriebssystemen ist dies meist kein Problem, weil in einem Adreßraum nicht mehrere Aktivitätsträger existieren
 - die *Lazy evaluation*-Techniken in Mach führen dazu, daß sehr häufig die pmap-Strukturen angepaßt werden müssen

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.33

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 VM auf Multiprozessoren (3)

- ▲ Problem
 - ◆ systeminitiierte Aktualisierung der pmap-Strukturen überschneidet sich mit der prozessorinitiierten Aktualisierung (z.B. Setzen des *Reference bit*)
 - ◆ inkonsistente Menge von TLB-Einträgen in der MMU möglich
 - ◆ Überschreiben der pmap-Strukturen durch die MMU möglich
- Mögliche Lösungen
 - ◆ Prozessoren werden durch Interrupt informiert und führen ein TLB-Konsistenz-Protokoll durch
 - ◆ die Nutzung geänderter Abbildungen wird verzögert, bis alle TLBs *gefushed* wurden
 - ◆ Inkonsistenzen werden in unkritischen Fällen toleriert (z. B. wenn die Zugriffsrechte erweitert werden)
 - ◆ Komplexere Algorithmen: z. B. *Shutdown* Algorithmus

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.35

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

J.5 Mach in verteilten Systemen

1 Netzwerk-Kommunikation

- Mach IPC nur für lokale Kommunikation konzipiert
 - ◆ Nachrichten können durch *Network-Server-Tasks* (im User-level!) an andere Knoten weitergeleitet werden
 - ◆ Capability-Funktionalität wird über Netzwerk abgebildet
 - ◆ Netzwerktransfer wird geschützt
 - DES-Verschlüsselung der Daten
 - Sequenznummern und Checksummen für Pakete
 - ◆ Authentisierung anderer Netzwerk-Server für den Capability-Transfer (*Secret Identifier*)

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.36

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Distributed Shared Memory

- der *Shared memory*-Bereich wird durch ein *Memory Object* repräsentiert
 - ◆ jede beteiligte Task bildet den Speicherbereich in ihren Adreßraum ab
 - ◆ dem *Memory Object* wird ein *External pager* zugeordnet
 - ◆ durch die *Lazy evaluation*-Technik werden die Seiten erst auf Anforderung in den Hauptspeicher (Seiten-Cache) geholt
 - ◆ im Fall von *Page faults* werden die Seiten von dem *External pager* angefordert
 - diese Anforderungen erfolgen über die regulären Mach Kommunikationsmechanismen, sind also auch auf Netzwerkkommunikation erweiterbar

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

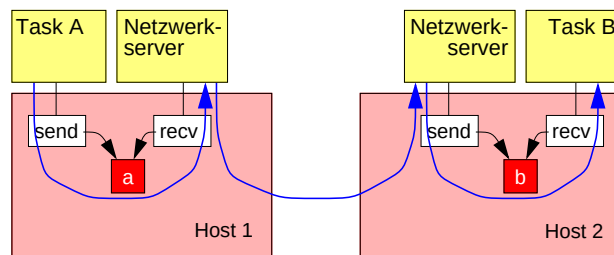
J-Mach.fm 1999-02-09 15:39

J.38

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Netzwerkkommunikation (2)

- Schaubild einer verteilungstransparenten Taskkommunikation



- ◆ Netzwerkserver fungieren als Vermittler
- ◆ sie stellen jeweils lokale Ports bereit und tunneln Kommunikation durch die Ports über ein Netzwerk

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.37

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Distributed Shared Memory (2)

- ◆ als *External pager* wird ein sog. **Shared memory server** benutzt, der Buch über das Caching der Seiten führt
- ◆ *read-only* Seiten können problemlos auf mehreren Knoten gecached werden
- ◆ bei Schreibzugriff wird ein *Page fault* erzeugt; der *Shared memory server* wird informiert, er entzieht den anderen Knoten den Zugriff auf die Seite und erlaubt dann den Schreibzugriff

SP I

Systemprogrammierung I

© Franz J. Hauck, Univ. Erlangen-Nürnberg, IMMD IV, 1998 / 1999

J-Mach.fm 1999-02-09 15:39

J.39

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.