MACH - virtuelle Speicherverwaltung

Eigenschaften

- portabel
- hardware-unabhängig
- unterstützt Multiprozessoren
  - ➥ parallel ablauffähig implementiert
- unabhängige Pager

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 phys. Seite gültig ist
    - *protection bits* – (z. B. *read/write, user/kernel*)
    - *modify bit*
    - *reference bit*

- typischerweise zwei mögliche Vorgehensweisen in heutigen Betriebssystemen zur Unterstützung von virtuellem Speicher

  - •• Mechanismen, die möglichst gut den Eigenschaften der verwendeten MMU entsprechen

    ➥ System nicht portabel (z. B. VMS für VAX)

  - •• Beschränkung auf Eigenschaften, die einfach auf jeder Hardware realisierbar sind

    ➥ System weniger leistungsfähig und ineffizient (z. B. UNIX in 4.3bsd)

Realisierung

Ziel

- möglichst flexible Unterstützung von virtuellem Speicher auf möglichst vielen Architekturen

Vorgehensweise

- Unterteilung der Implementierung

  ➥ 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

➥ 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 Segment-Registern 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 (vgl. herkömmliche VM-Implementierungen - z. B. VAX)

- VM-unabhängige Pager

  •• Seiten-Ein- und Auslagerung wird durch VM von Pagern über Messages angefordert

- 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

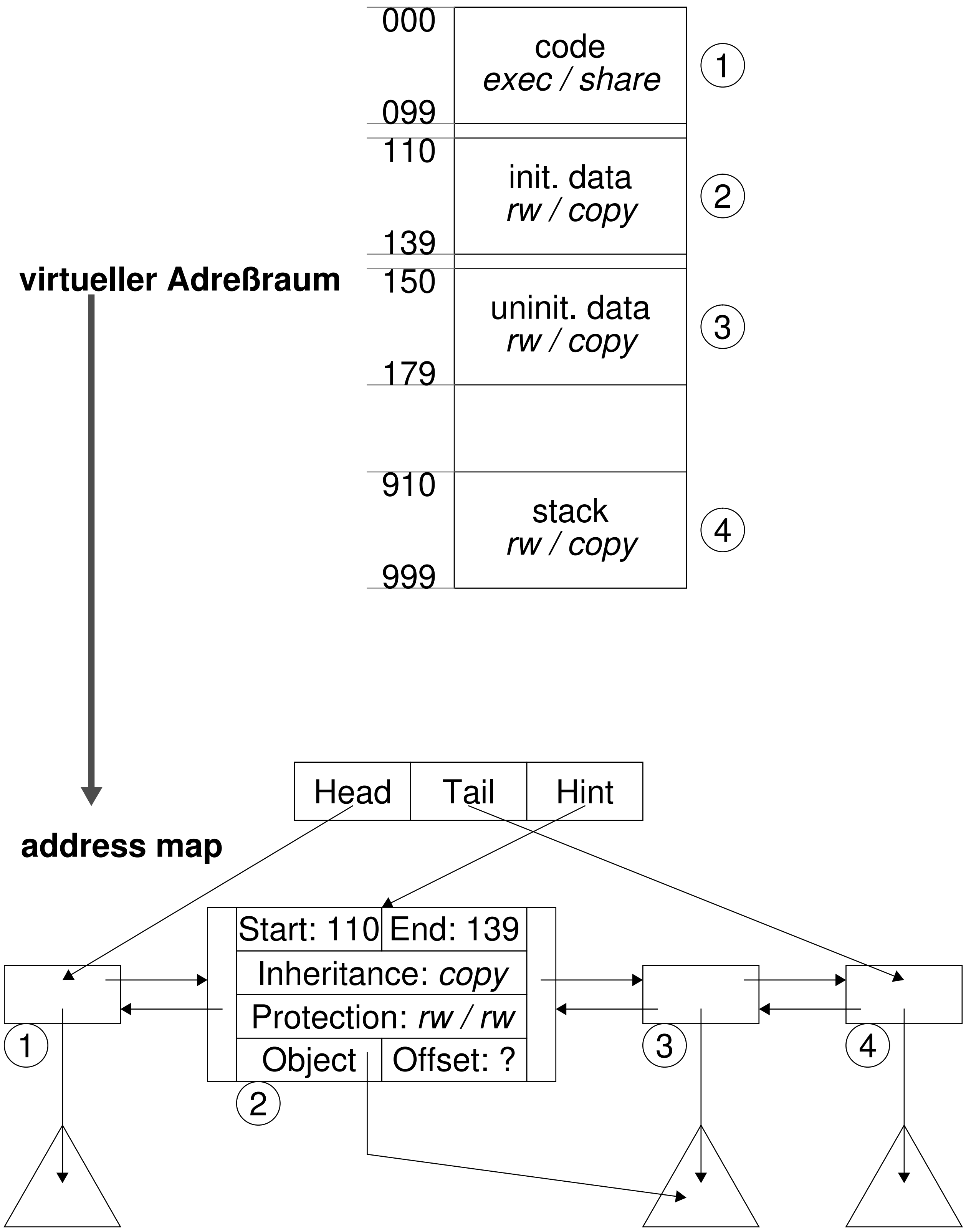
  •• Copy-on-write

Datenstrukturen

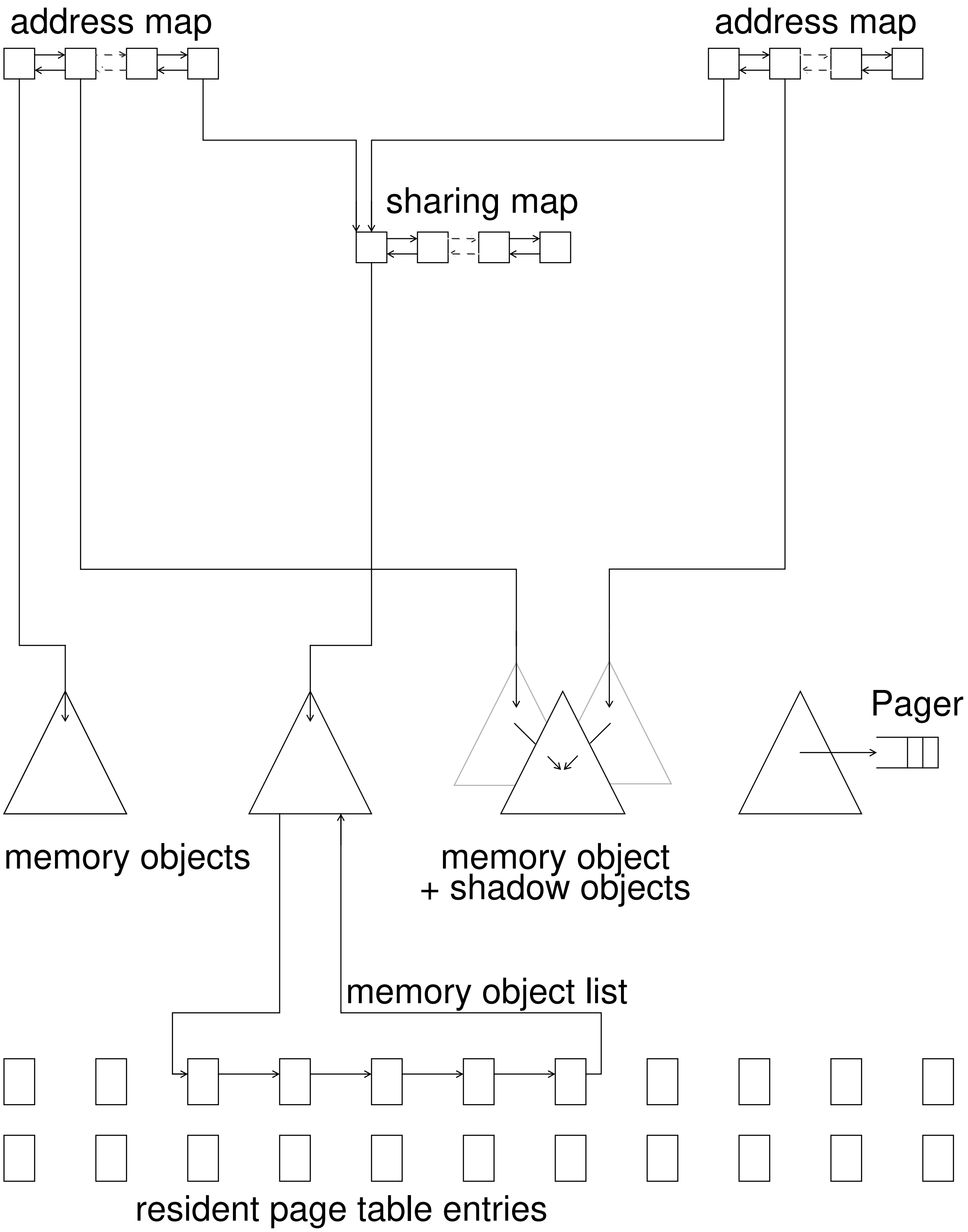
- Address Map
  - Abbildung von (virtuellen) Speicherbereichen mit gleichen Eigenschaften (Schutz, Vererbung) in einen Bereich eines *Memory Objects*
  - als verkettete Liste von *address-map-entries* realisiert
  - ein *address-map-entry* enthält:
    - Vorwärts- und Rückwärts-Verzeigerung
    - Start- 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 *Memory Object*
- Sharing Maps
  - zur Realisierung von shared Memory
  - Zwischenstufe zwischen *address map* und *Memory Object*
  - um Änderungen im *address map* -Eintrag eines Objekts nicht in den *address maps* mehrerer Tasks nachtragen zu müssen

- Memory Object
  - Bindeglied zwischen virtuellem Adreßraum und Hintergrundspeicher
  - weiß für eine Anzahl Bytes, wie sie zu beschaffen sind und was mit ihnen im pagout-Fall zu tun ist
  - Pager (intern oder extern) realisiert Ein-/Auslagerung von Seiten
  - *Memory Objekt* hat Information, welche seiner Daten gerade im Hauptspeicher *gecached* sind
- Shadow Object
  - zur Realisierung von *copy-on-write*
  - nimmt Seiten auf, die kopiert werden mußten
  - verweist für unveränderte Seiten auf das original-Objekt
- Resident Page Table
  - Verwaltung des **Hauptspeicher-Caches** für *Memory-Object*-Seiten
  - Verkettung in Listen (*memory object, memory allocation, object/offset hash*)

- Datenstrukturen: virtueller Adreßraum - address map



- Datenstrukturen - Gesamtüberblick



Benutzerschnittstelle

MACH-VM stellt folgende Funktionen zur Verfügung:

- virtuelle Speicherbereiche allokieren 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 VM anderer Tasks (Debugger!)
- Status- und Statistikabfragen