Hierarchische Dateisysteme

Daniel Laffling

Seminarvortrag
Ausgewählte Kapitel der Systemsoftware
Wintersemester 2015/2016

11. Januar 2016

daniel.laffling@fau.de

Department of Computer Science IV
Distributed Systems and Operating Systems
Friedrich-Alexander University Erlangen-Nuremberg

Überblick

- Motivation
- Konzept
- Implementierung
- Hürden und Verbesserungen
- Einflüsse auf heutige Systeme
- Fazit

Motivation

- Ausgangssituation
  - Mitte der 1960er Jahre
  - komplexe Rechensysteme oft in hoch-kritischen Einsatzbereichen
  - Mehrbenutzersysteme ohne time-sharing
  - Datenverwaltung auf Sekundärspeichern obliegt Programmierer
  - Werkzeuge zur Datenverwaltung spärlich/offline/nicht vorhanden
- Risiken
  - Programmierfehler
  - Systemfehler
  - unberechtigter Zugriff
  - (ungewollte) Manipulation
- Schadenausmaß unvorhersehbar

Motivation

- Multics soll Abhilfe schaffen
  - Sicherheit
  - Benutzerfreundlichkeit
  - Performanz
  - Portabilität & Erweiterbarkeit

Konzept

- Daley & Neumann: A general-purpose file system for secondary storage

- Konzept einer mächtigen Datenverwaltungssoftware für Multics

- Verwaltung durch Baumstruktur aus
  - Dateien
  - Verzeichnissen
  - Verknüpfungen

Konzept

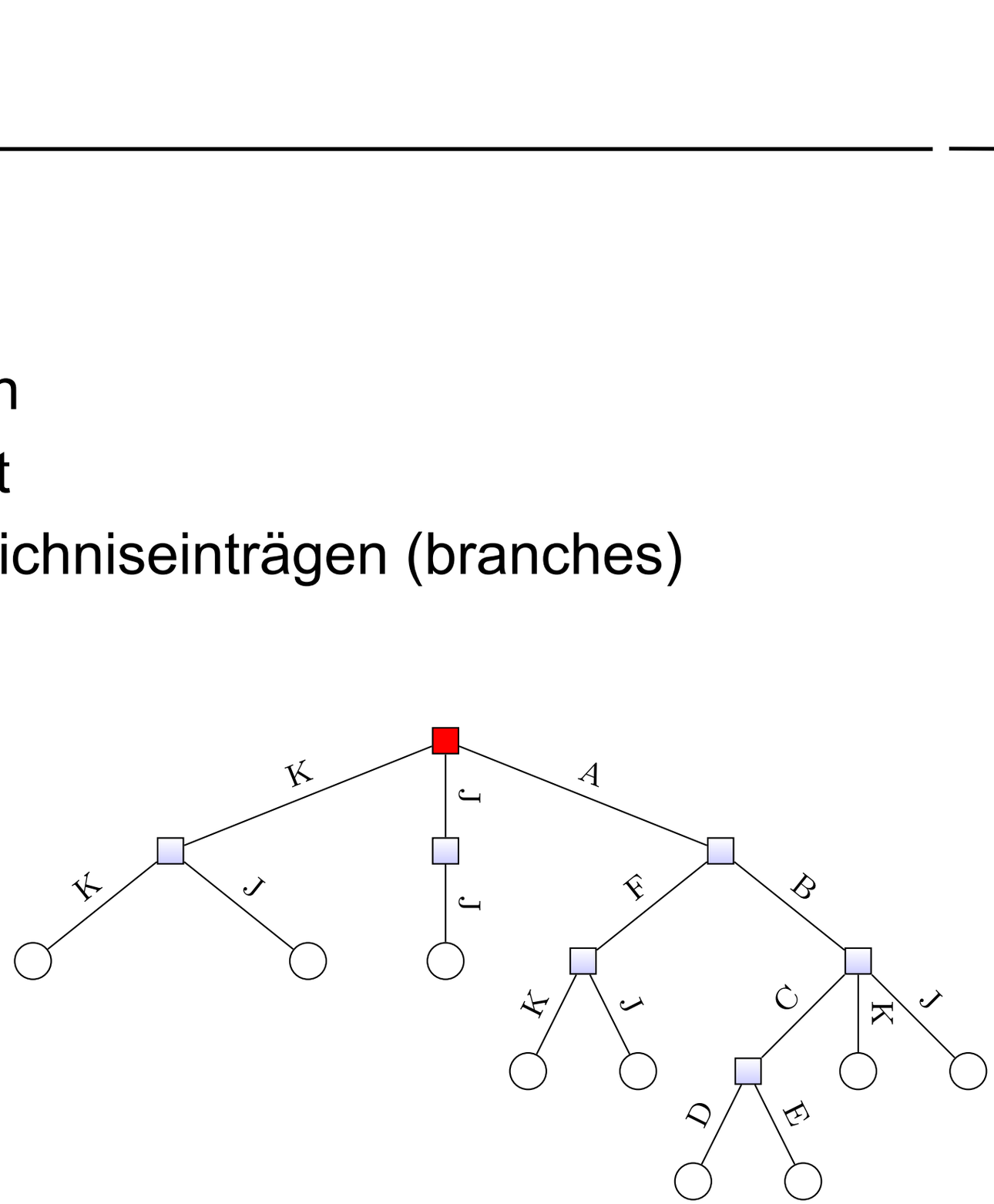
- Datei
  - geordnete Sequenz von Elementen
  - Element implementierungsabhängig
    - Maschinenwort
    - Buchstabe
    - Bit
  - Modifikation nur durch Dateisystemschnittstelle
  - Auf Dateisystemebene formatlos
  - Adressierung von Elementen durch Index relativ zum Anfang der Datei
  - frei wählbarer symbolischer Name
  - Kapselung eines logisch zusammengehörenden Datensatzes
  - Segment im Speicher

Konzept

- Verzeichnis
  - spezielle Datei
  - trägt symbolischen Namen
  - von Dateisystem verwaltet
  - beinhaltet Liste von Verzeichniseinträgen (branches)
    - symbolische Namen
    - systemrelevante Attribute

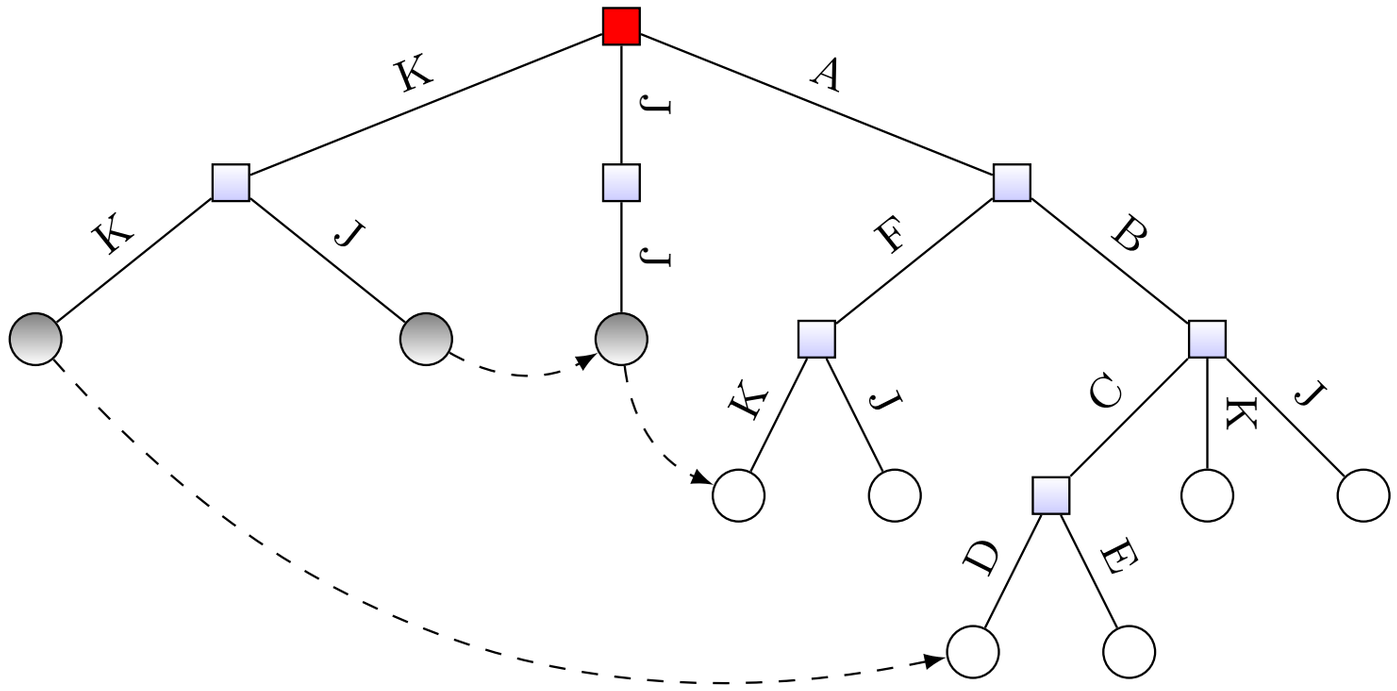
- Verzeichniseinträge
  - reguläre Dateien
  - spezielle Dateien

- Ein Wurzelverzeichnis



Konzept

- Verknüpfungen
  - spezielle Datei
  - beinhaltet keine Daten, sondern Verkettung symbolischer Namen zum eigentlichen Ziel



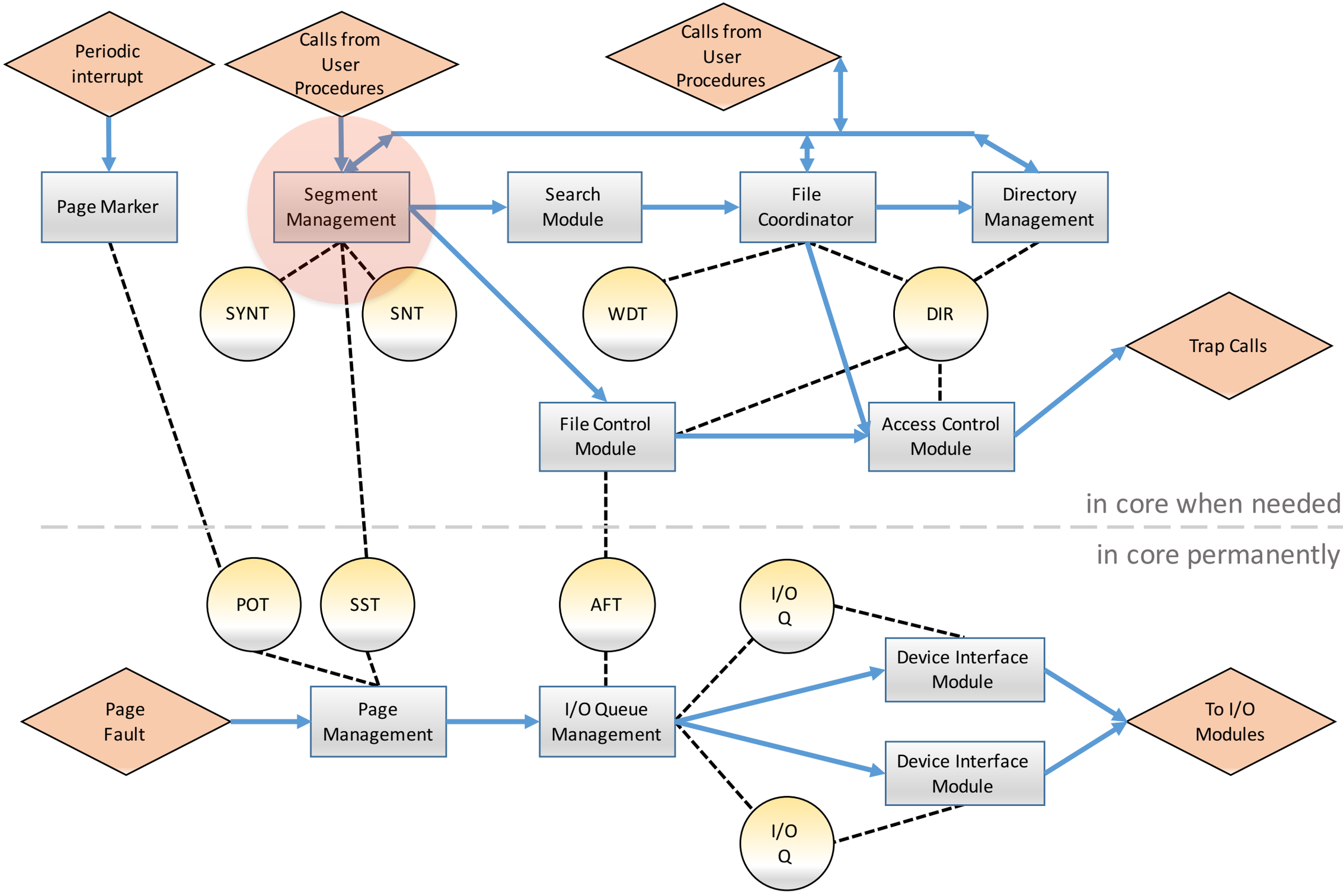
Konzept

- Dateiattribute
  - in Liste der Verzeichniseinträge
  - enthält
    - Zeitpunkt der Erstellung
    - Zeitpunkt der letzten Modifikation
    - Größe der Datei (bei Verzeichnissen unbenutzt)
    - Zugangsbeschränkungen für Benutzer
    - aktueller Status der Datei (lesende Zugrife, schreibende Zugriffe, teilende Zugriffe)

Implementierung

- Struktureller Aufbau des Dateisystems
  - mehrere feingranular aufgeteilte Module und Informationsdatenbanken
    - prozedurlokale Daten
    - globale Daten
  - Grundprinzip: ***Everything is a segment***

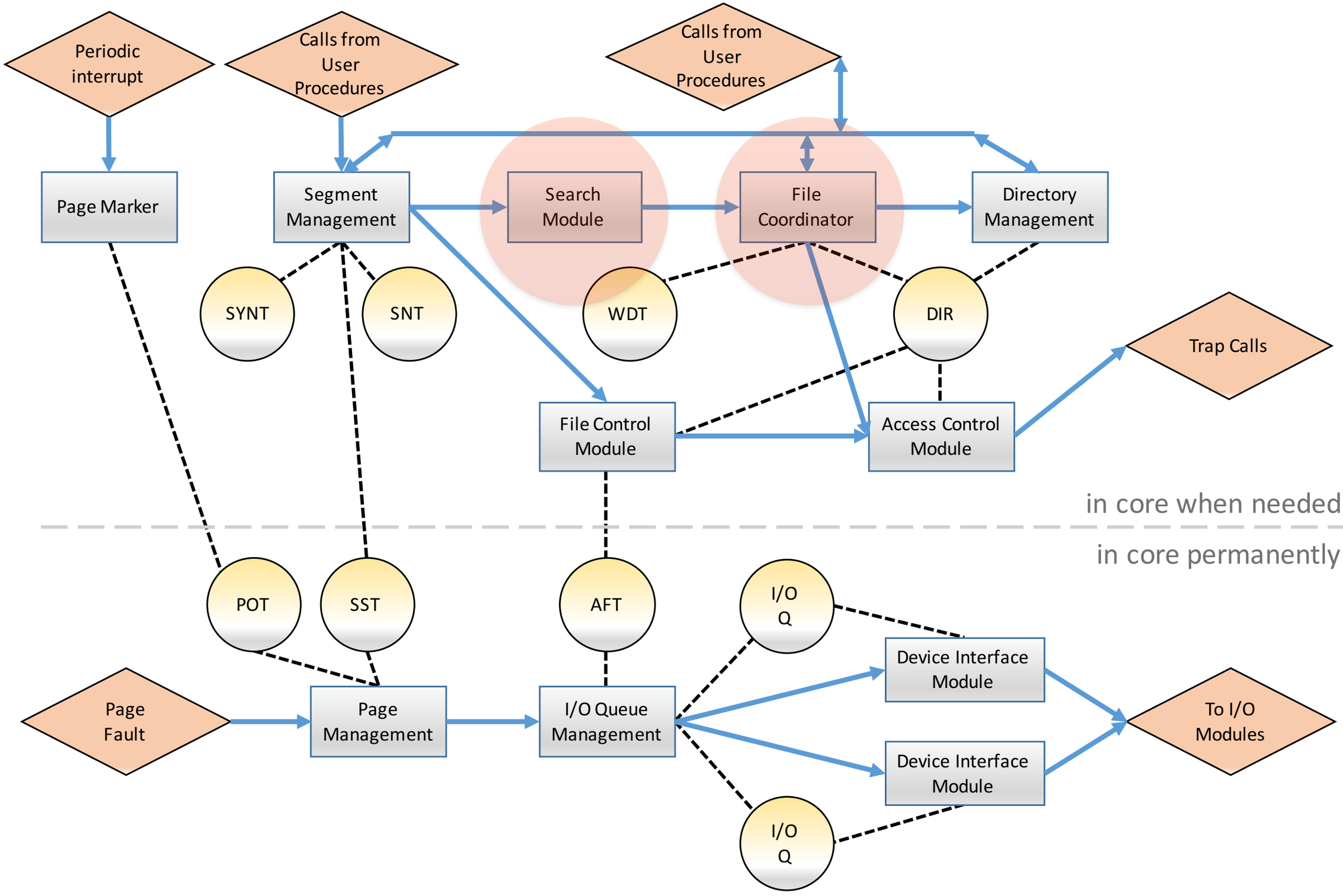
Implementierung



Implementierung

- Segment Management Modul
  - Segmentverwaltung für laufenden Prozess
  - führt prozedurlokale Liste aller bis dato bekannten Segmente eines Prozesses (Segment Name Table)
    - call name: Symbolischer Name des Segments
    - tree name: beinhaltender Verzeichnispfad
    - Segmentnummer im Speicher
  - überprüft Aktivität eines Segments in globaler Segment Status Table
    - aktiv: Segment befindet sich im Hauptspeicher
    - inaktiv: Segment befindet sich nicht im Hauptspeicher
  - Löst bei Inaktivität eines Segments Ausnahmeprozedur aus
    - unbekannt: Auffindung und Einlagerung (Search Module)
    - inaktiv: Wiedereinlagerung

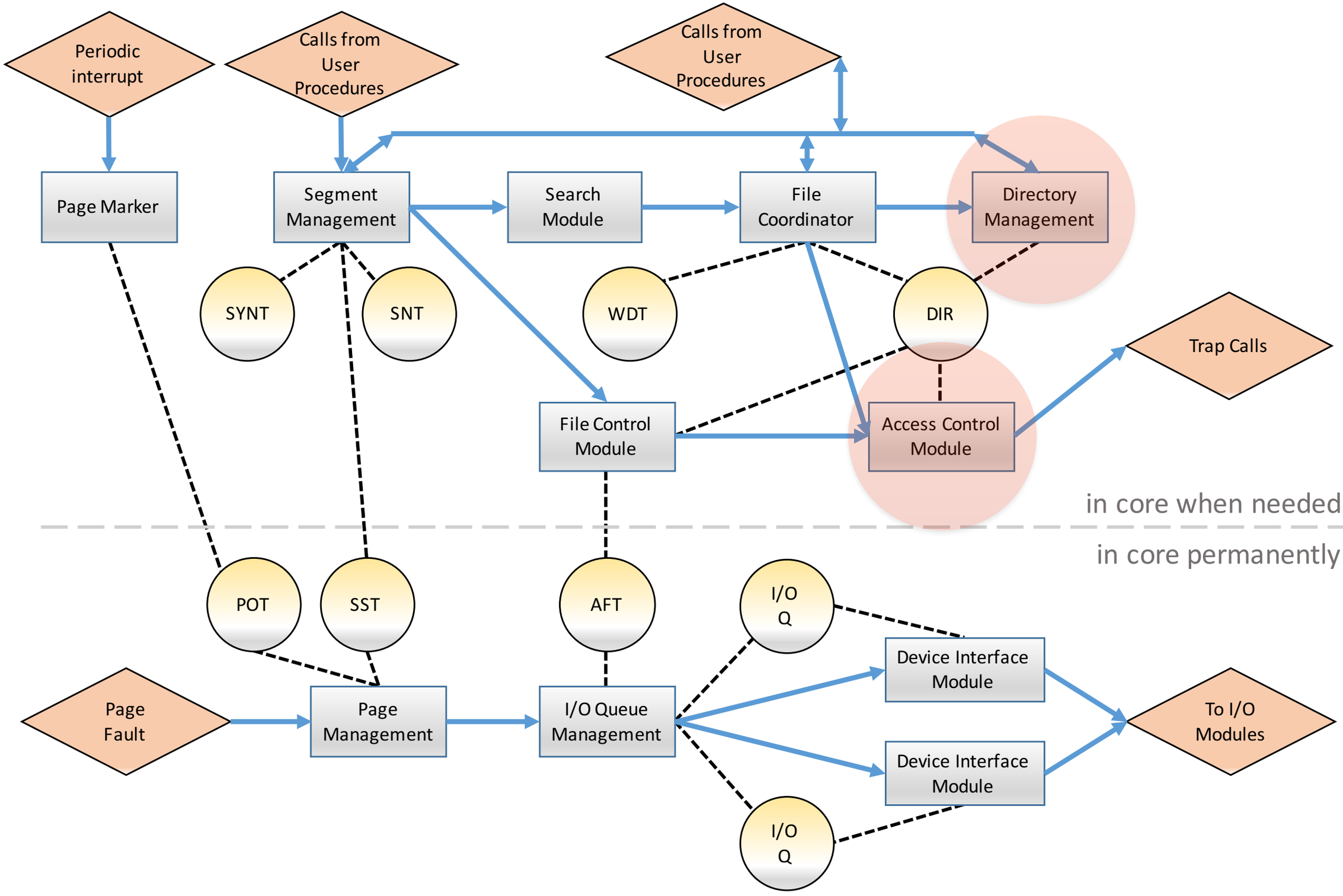
Implementierung



Implementierung

- Search Module
  - Modul zur Auffindung unbekannter Segmente im Sekundärspeicher
  - traversiert Verzeichnisbaum
  - braucht für Suche im Verzeichnis File Coordinator Module
- File Coordinator Module
  - Erzeugung und Entfernung von Verzeichniseinträgen
  - Abfrage von Statusinformationen
  - Modifikation von Zugangsbeschränkungen
  - Wechsel des Arbeitsverzeichnisses
  - liest gültiges Arbeitsverzeichnis für jeden Prozess aus globaler Working Directory Table
  - beauftragt Directory Management mit der Suche in einem Verzeichnis

Implementierung

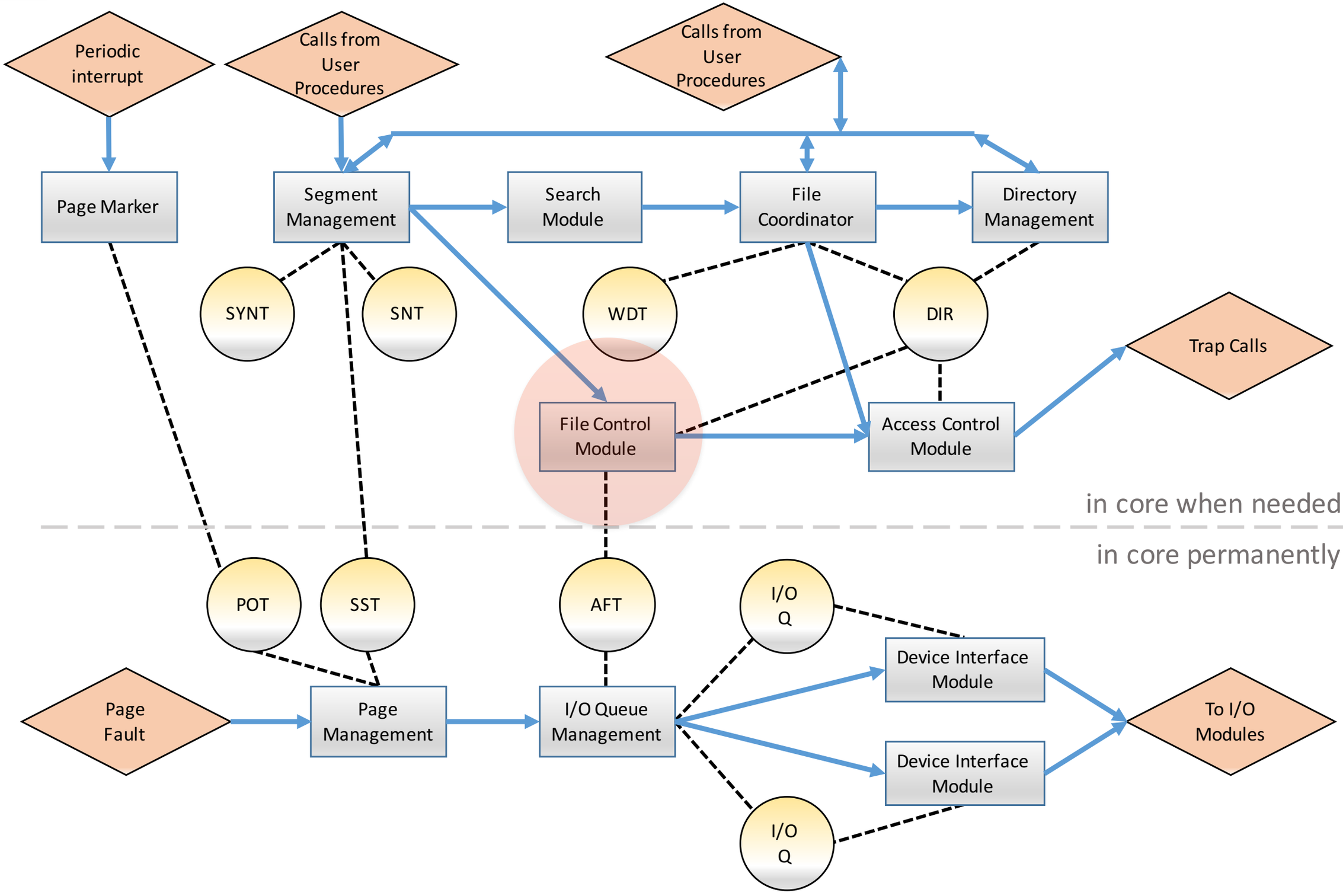


Implementierung

- Directory Management Module
  - Durchsuchen eines Verzeichnisses
  - Auslagerung einer erwarteten Rekursion
    - bekommt für die Suche nur Verzeichnispfad (tree name)
    - fragt Segment Management nach Segmentnummer zu Verzeichnispfad
    - falls Segment inaktiv folgt Ausnahmeprozedur zur Wiedereinlagerung
    - Wiedereinlagerung durch Suche des tree names im übergeordneten Verzeichnis
    - Worst case: Rekursion bis zum Wurzelverzeichnis (permanent eingelagert)

- Access Control Module
  - Abfrage von Zugangsberechtigungen
  - erhält Information über Benutzer und Art des beabsichtigten Zugriffs
  - Auswertung der Rechte
  - Rückgabewert über Auslösung einer Ausnahmebehandlung, auf welche durch den Aufrufer reagiert werden kann (Akzeptanz/Modifikation)

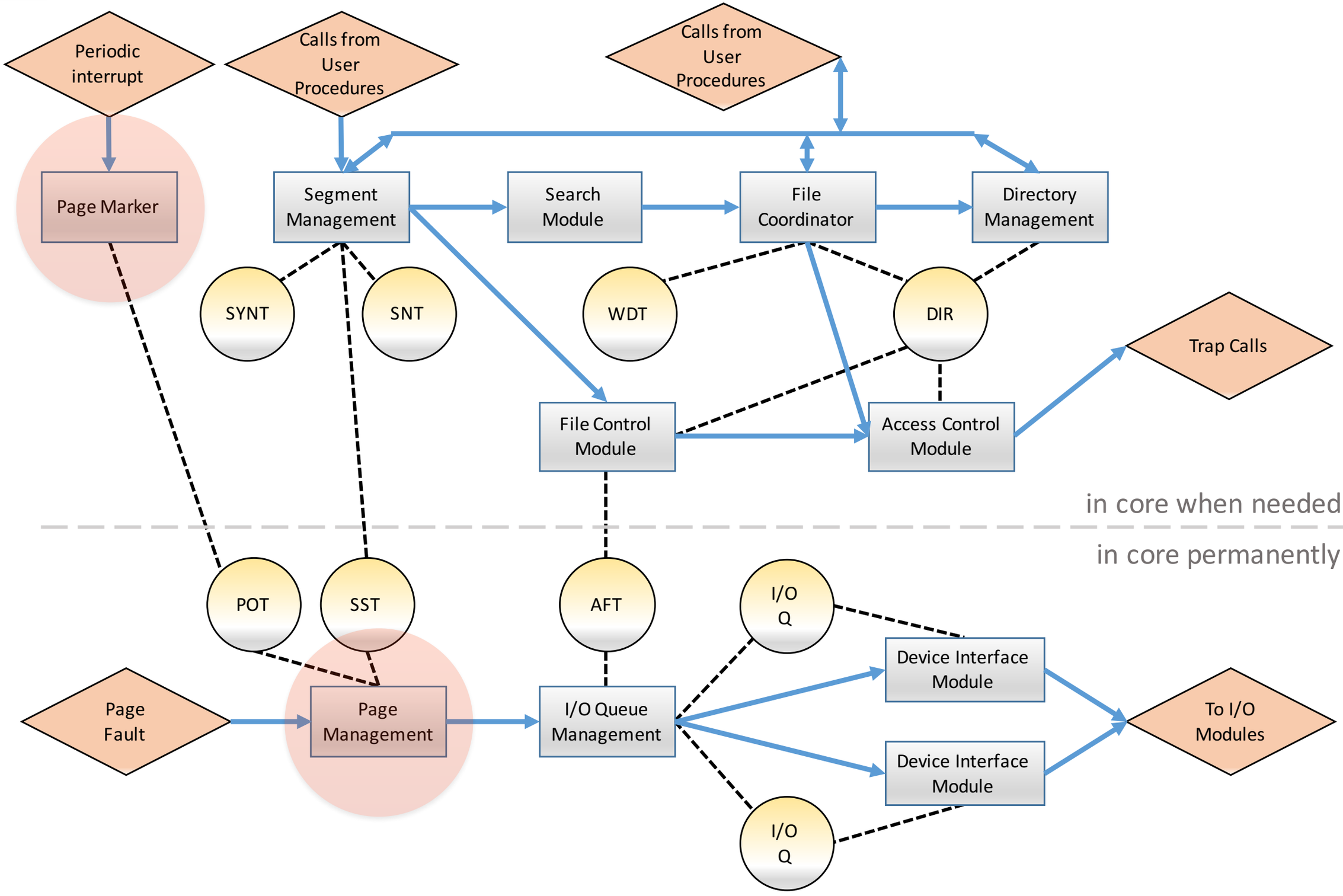
Implementierung



Implementierung

- File Control Module
  - Untermodul von Segment Management
  - Öffnen und Schließen von Dateien
  - verwaltet globale Active File Table
    - alle im System geöffneten Dateien incl. Zugriffszahlen
      - lesend: alle gleichzeitig möglich
      - schreibend: max 1 Prozess gleichzeitig
      - teilend: mehrere Prozesse lesend/schreibend möglich (Interprozesskommunikation)

Implementierung



Implementierung

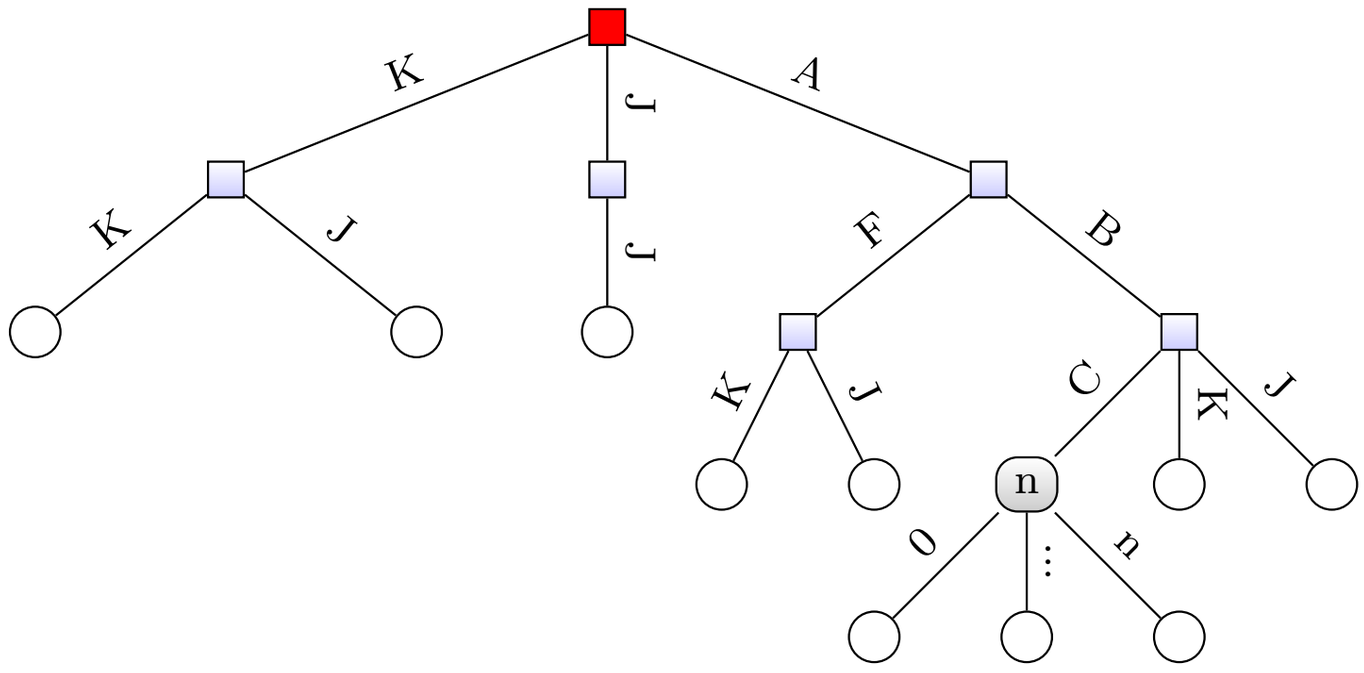
- Page Marker
  - periodisch aufgerufene Interruptroutine
  - Markiert potenziell ungenutzte Seiten durch Eintrag in Page Out Table
- Page Management
  - Bei Page Fault (neue Speicherseite/Speicherseite aus bestehender Datei) Verdrängung des ersten Eintrags der Page Out Table aus dem Speicher
  - Aktualisierung der Segment Status Table

Implementierung

- Weitere Module
  - Eigenständige Backup & Restore Routinen
    - inkrementell
    - wöchentlich
    - kontinuierliche Abgleiche mit Backups
  - Datenpriorisierung
    - automatische Verlagerung von Dateien nach Zugriffshäufigkeit auf schnellere/langsamere Speichermedien
  - Hintergrundmodule zum Einlesen und Auslesen von Daten
    - File to Card
    - Card to File
    - File To Tape
    - …

Hürden und Verbesserungen

- Was wurde durch initiales Konzept nicht beachtet?
  - Dateigrößen
    - Ursprungskonzept beschränkt auf ca. 1MiB pro Segment/Datei
- Lösung: Multi Segment Files
  - Repräsentation als spezielles Verzeichnis
    - Verzeichnisinhalt: Aufsteigend nummerierte Dateien
    - Kenntlichmachung durch Setzen des Größe-Attributes auf die Anzahl der Verzeichnisinhalte



Hürden und Verbesserungen

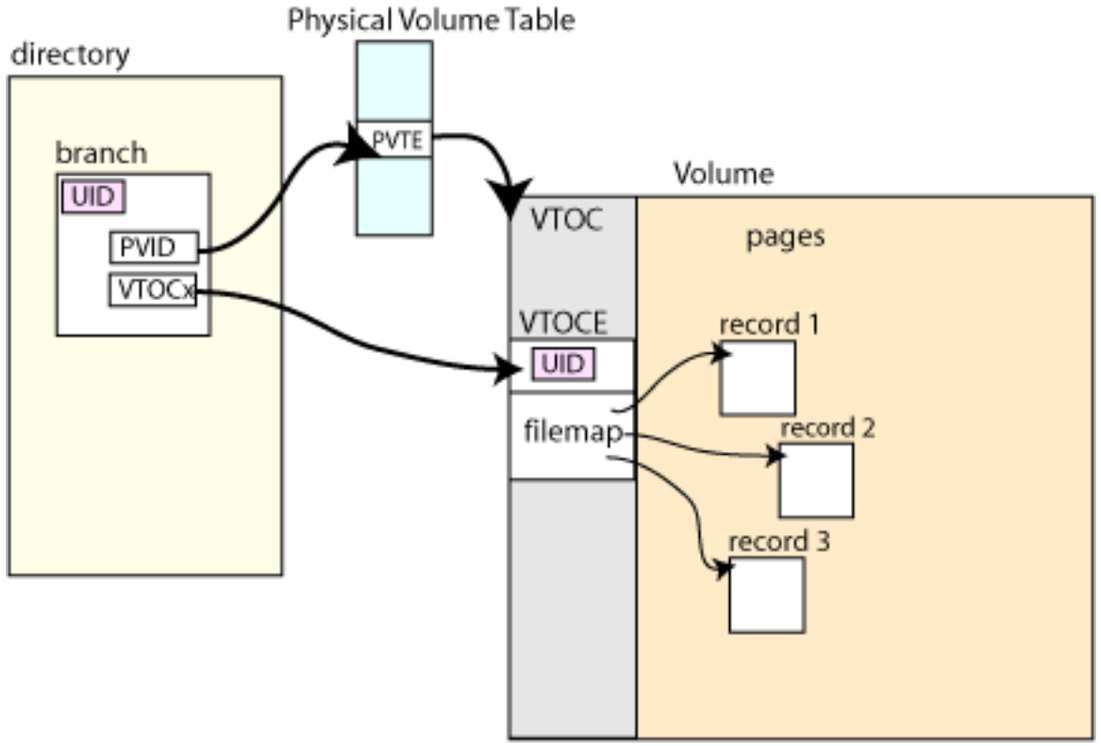
- Was wurde durch initiales Konzept nicht beachtet?
  - inkonsistente Daten
    - Alle Sekundärspeicher zu einem logischen Datenträger vereint
      - gleichmäßigere Nutzung von Speichern (striping)
    - Defekt in einem Speichermedium hat Rekonfiguration des gesamten logischen Verbundes zur Folge
      - Dauer: mehrere Stunden bis Tage
  - Laufwerkskapazitäten
    - Pages im Sekundärspeicher fortlaufend adressiert
    - 18 Bit Adresse führt zur Erreichung des Gesamtspeichermaximums
  - Austauschbare Speichermedien
    - Benutzerbedürfnis
    - auf anderen Maschinen u.a. von IBM möglich

Hürden und Verbesserungen

- Lösung durch Reformation des Dateisystems 1973
  - Beibehaltung der Benutzerschnittstelle
  - Separation zwischen logischer und physikalischer Speicherung
  - Vermeidung von Rekonfigurationen auf dem Gesamtspeicher
  - Unterstützung von austauschbaren Speichern

Hürden und Verbesserungen

- Umsetzung
  - Aufbrechen des generellen Zusammenschlusses aller Speicher zu einer Gesamtheit.
  - Laufwerke erhalten eigene Laufwerksinhaltstabelle
  - Verzeichniseinträge erhalten Information über ihren Speicherort



Einflüsse auf heutige Systeme

- UNIX
  - Maßgeblich Entwickelt von Dennis Ritchie und Ken Thompson (Bell Labs)
  - Beide gehörten zur Dateisystemgruppe bei der Multics-Entwicklung
  - Sehr ähnliche Dateiaufbewahrung zu Multics schon von Beginn an
- RAID 0
  - „Redundant arrays of inexpensive disks“ (Patterson et al.1988)
  - RAID 0
    - Striping
    - eigentlich kein RAID
    - Grundprinzip sehr ähnlich zur Multics Basisimplementierung

Einflüsse auf heutige Systeme

- Intel Turbo Memory (2007)
  - kleine schnelle PCIe Flash Speicherbausteine zur Verwendung mit Windows Vista
  - Prinzip: Vorhalten von hoch frequentierten Daten auf schnellem Speicher (Bootcode, etc.)
  - teuer
  - Geschwindigkeitsvorteil im Realbetrieb nicht ausreichend im Vergleich zu Preis
  - hat sich demzufolge nicht durchgesetzt
- Apple FusionDrive (2012)
  - Beruht auf Hystor Konzept von Intel von 2011
  - Zusammenschluss eines Solid State Drives mit einer gewöhnlichen Magnetplatte.
  - Prinzip: Vorhalten von hoch frequentierten Daten auf schnellem Speicher (Bootcode, etc.)
  - Benutzer hat keine Eingriffsmöglichkeiten.
  - Prinzipiell bereits von Beginn an Multics-Feature

Fazit

- Grundbedürfnisse an Speicherverwaltung unverändert
- immer noch weitgehend hierarchische Speicherung von Daten auf nahezu allen Systemen
- Dateisysteme immer noch Abstraktionsschicht zwischen Betriebssystem und Benutzer
- Konzepte von 1965 wegweisend für Entwicklungen der darauffolgenden Zeit bis heute

Vielen Dank für die Aufmerksamkeit

FRAGEN?

Quellen

- http://multicians.org - aufgerufen am 02.01.2016.
- F. Chen, D. A. Koufaty, and X. Zhang. Hystor: making the best use of solid state drives in high performance storage systems. In Proceedings of the international conference on Supercomputing, pages 22–32. ACM, 2011.
- R. C. Daley and P. G. Neumann. A general-purpose file system for secondary storage. In Proceedings of the November 30–December 1, 1965, fall joint computer conference, part I, pages 213–229. ACM, 1965.
- P. Green. Multics virtual memory: Tutorial and reflections. Retrieved January, 29:2001, 1993.
- D. A. Patterson, G. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (RAID), volume 17. ACM, 1988.
- D. M. Ritchie and K. Thompson. The unix time-sharing system. Communications of the ACM, 17(7):365–375, 1974.
- T. V. Vleck. The new storage system. 1995.
- Bisherige Seminarvorträge AKSS-WS15 https://www4.cs.fau.de/Lehre/WS15/MS_AKSS/