File Service Design

SDI Praktikum

Katja Renner, Philipp Kirchhofer

Karlsruher Institut für Technologie

10. Juni 2010

Gliederung

Motivation & Designrichtlinien

Motivation Dateien & Dateisysteme

Kurze Wiederholung aus SysArch

- Große Menge an Daten speichern
- Daten dauerhaft speichern
- Daten einfach wiederfinden

Motivation VFS

Anwendungen können auf Dateien zugreifen und diese verändern, ohne den konkreten Speicherort (Festplatte, CD, Netzwerk) zu kennen.

Zentraler „Ansprechpartner“ für Anwendungen

Motivation & Designrichtlinien

Designrichtlinien

- Baumbasierte Verzeichnisstruktur
- Möglichst wenig IPC Nachrichten
  - 2 Aufruftypen für File Service
- Stateful Server
  - Gemeinsam genutzte Speicherbereiche für VFS und FS
- Sinnvolle (?) Erweiterungen der POSIX Norm
  - Keine Überlappung von Lese- und Schreibzugriffen
  - Persistente Dateipositionen

File Service Aufruftypen

1. Rekursiver Aufruf über VFS
2. Direkter Aufruf FS

$\Rightarrow$ Reduzierung der Anzahl an IPC Nachrichten

Motivation & Designrichtlinien

POSIX $\Leftrightarrow$ L4

Bibliothek für Programme stellt Standard POSIX Funktionen zur Verfügung.
Wandelt Aufrufe in L4 IPC Nachrichten um.

Vorteile:

- Standard im UNIX Bereich
- Ermöglicht einfache Programmierung
- Portabilität
- Ausgereifte Spezifikationen existieren

Motivation & Designrichtlinien

Beispiel: POSIX Dateiaufrufe

int open(const char *pathname, int flags);

Öffnen einer Datei

Flags:

- Read only
- Write only
- Read and Write
- Persistent Position (Erweiterung)
- Create File
- Append

Motivation & Designrichtlinien

Beispiel: POSIX Dateiaufrufe

int close(int fd);

Datei schließen

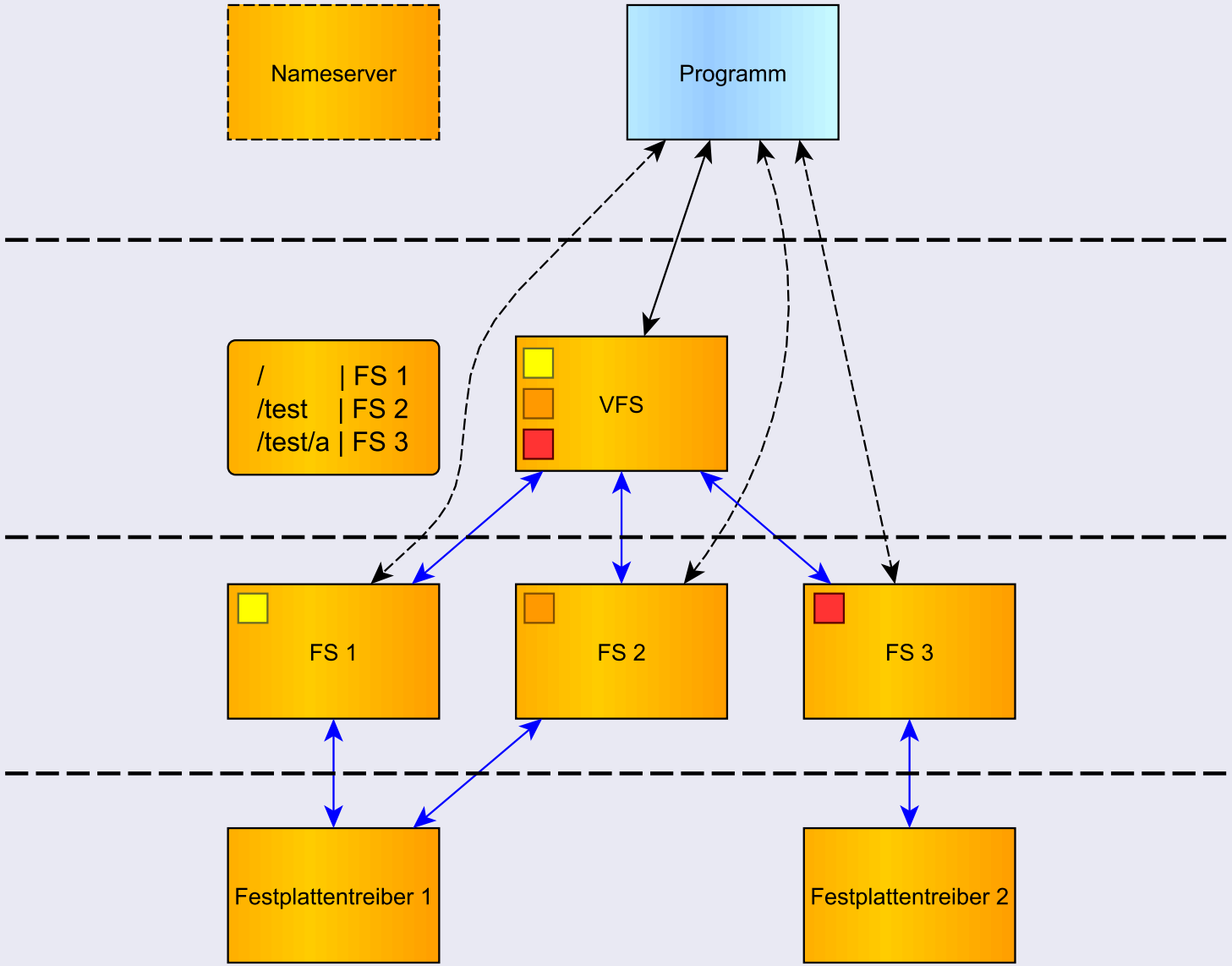
size_t read(int fd, void *buf, size_t count);

Daten lesen

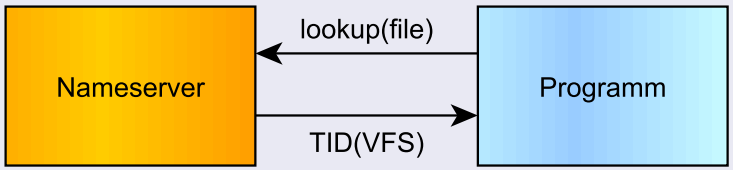
size_t write(int fd, const void *buf, size_t count);

Daten schreiben

Server Kommunikation - Übersicht



Server Kommunikation - Lookup des VFS



Server Kommunikation - Öffnen von Dateien (Aufruftyp 1)

Gemeinsam genutzte Offene Dateien Tabelle

Handle[a]	FS Handle[b]	Flags	Position	Thread ID

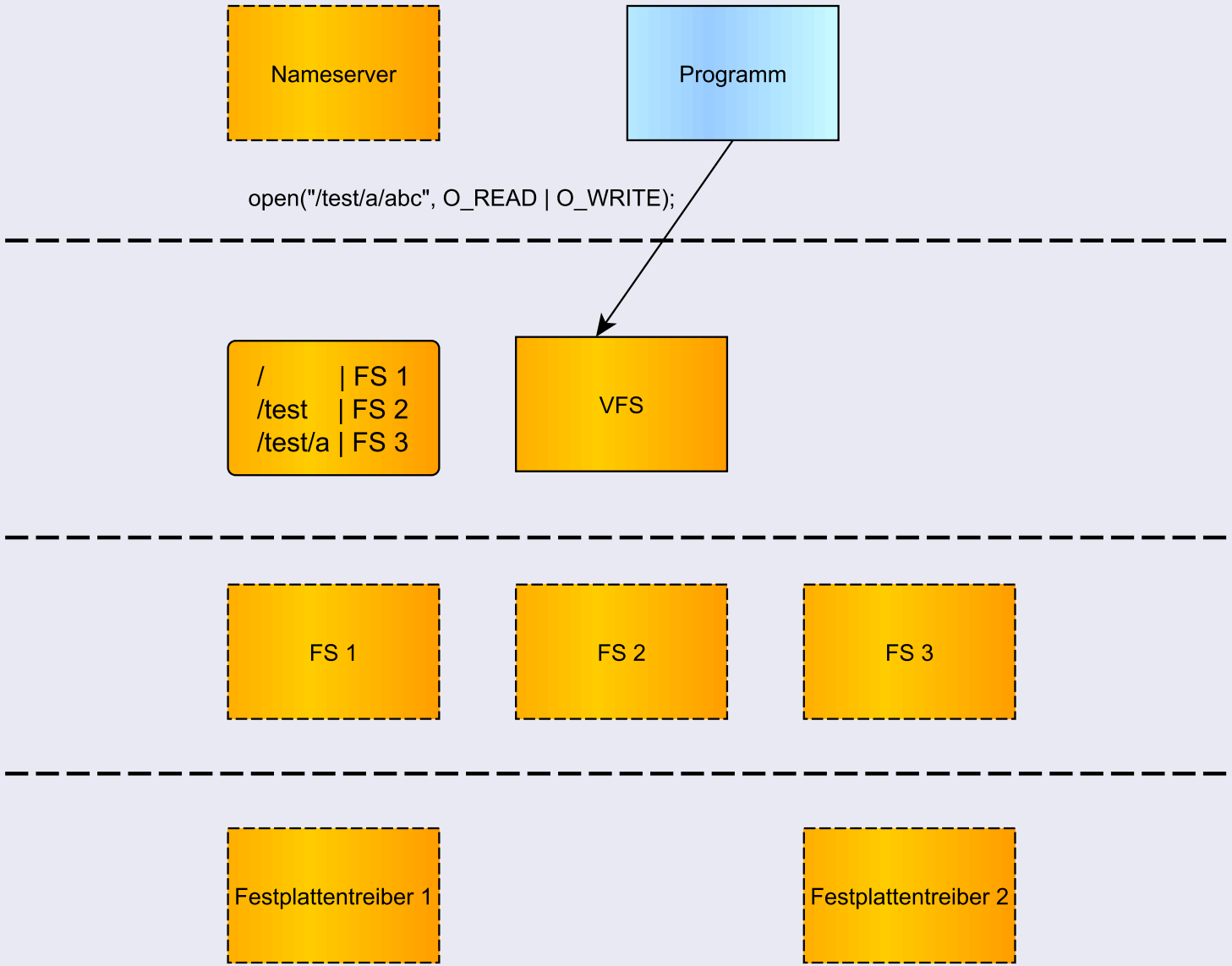
Flags

Read only, Write only, Read and Write, Persistent Position

[a] eindeutig

[b] FS privat

Server Kommunikation - Öffnen von Dateien (Aufruftyp 1)

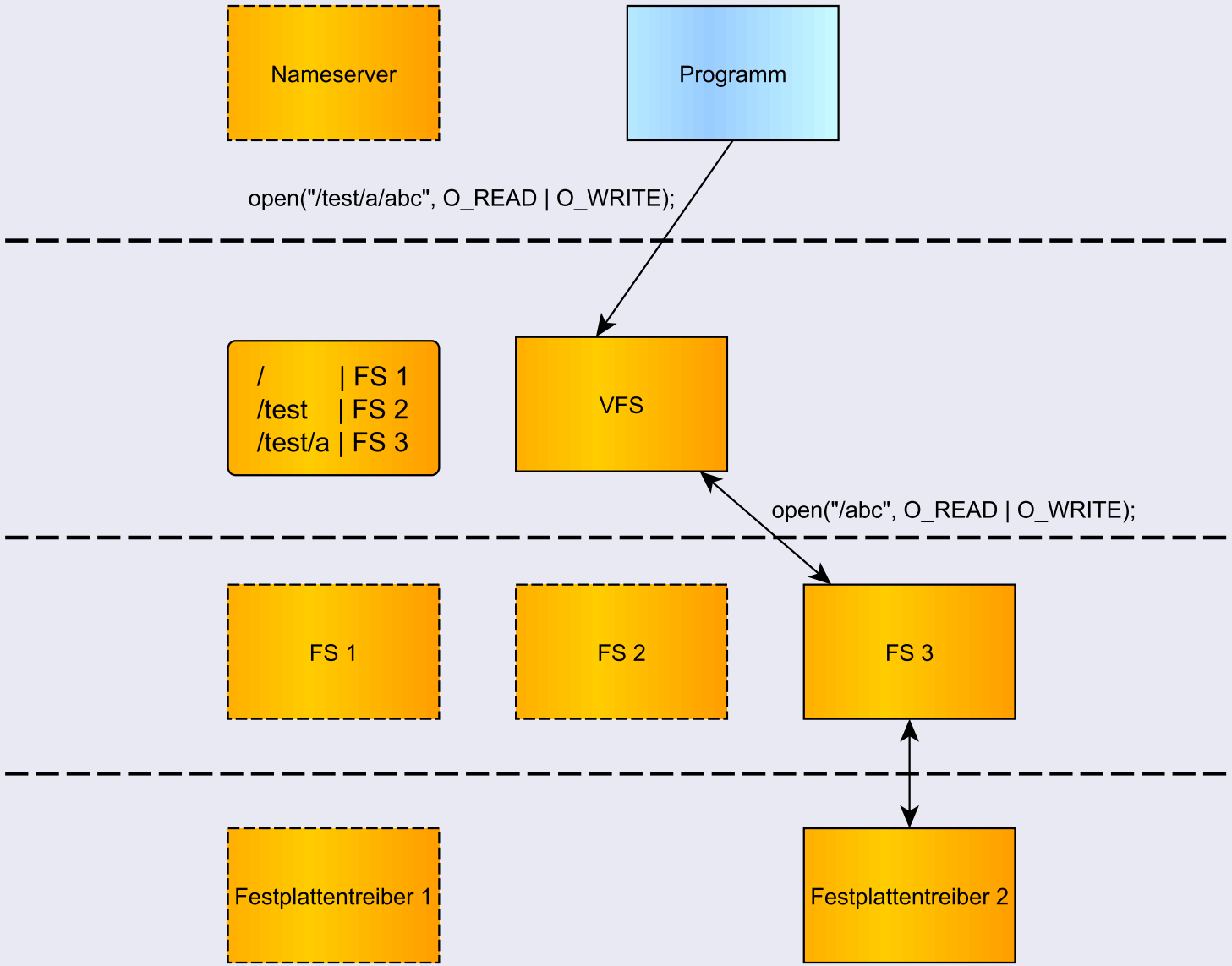


Server Kommunikation - Öffnen von Dateien (Aufruftyp 1)

Gemeinsam genutzte Offene Dateien Tabelle

Handle	FS Handle	Flags	Position	Thread ID
0xAB		Read and Write	0	0xC0FFEE

Server Kommunikation - Öffnen von Dateien (Aufruftyp 1)

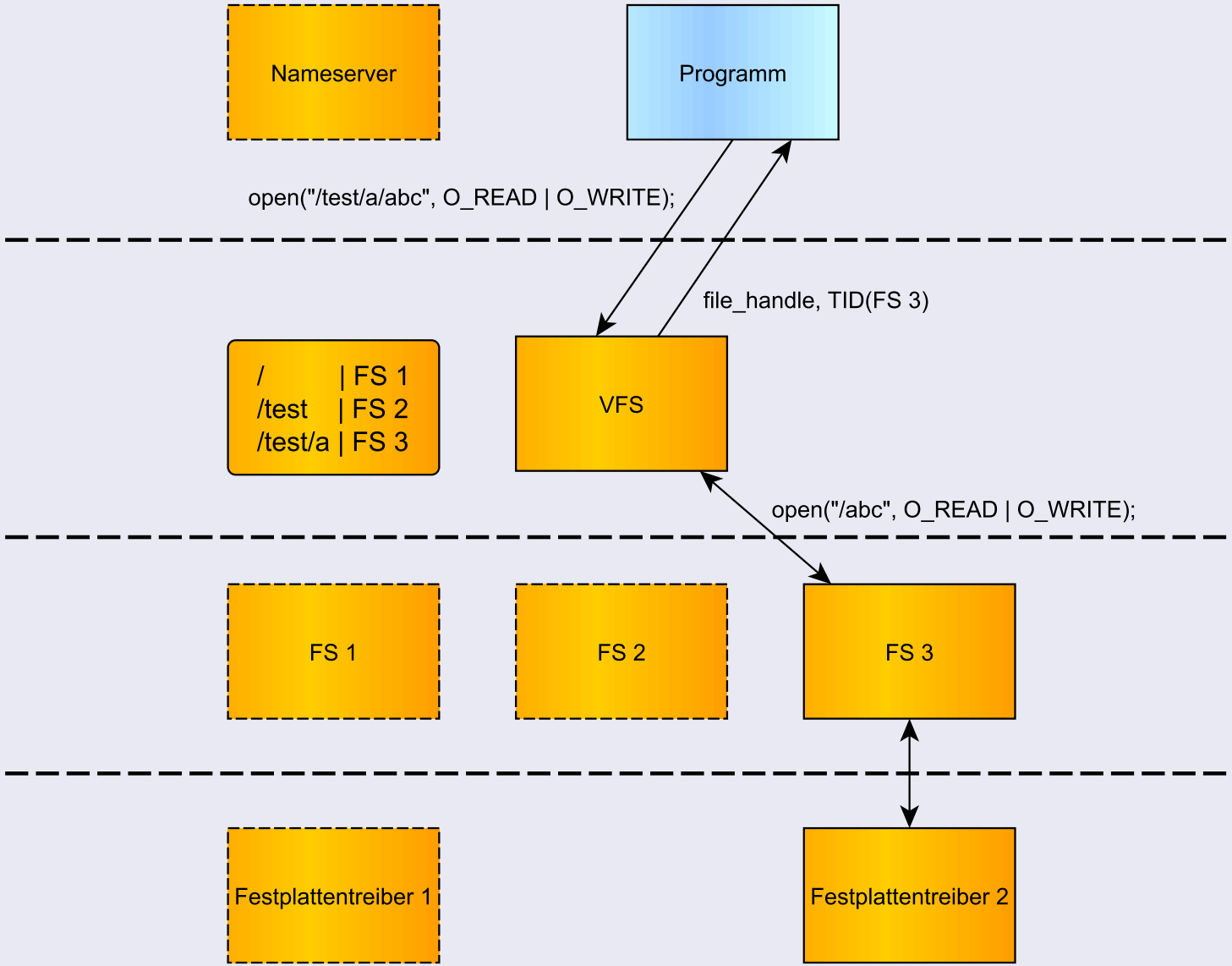


Server Kommunikation - Öffnen von Dateien (Aufruftyp 1)

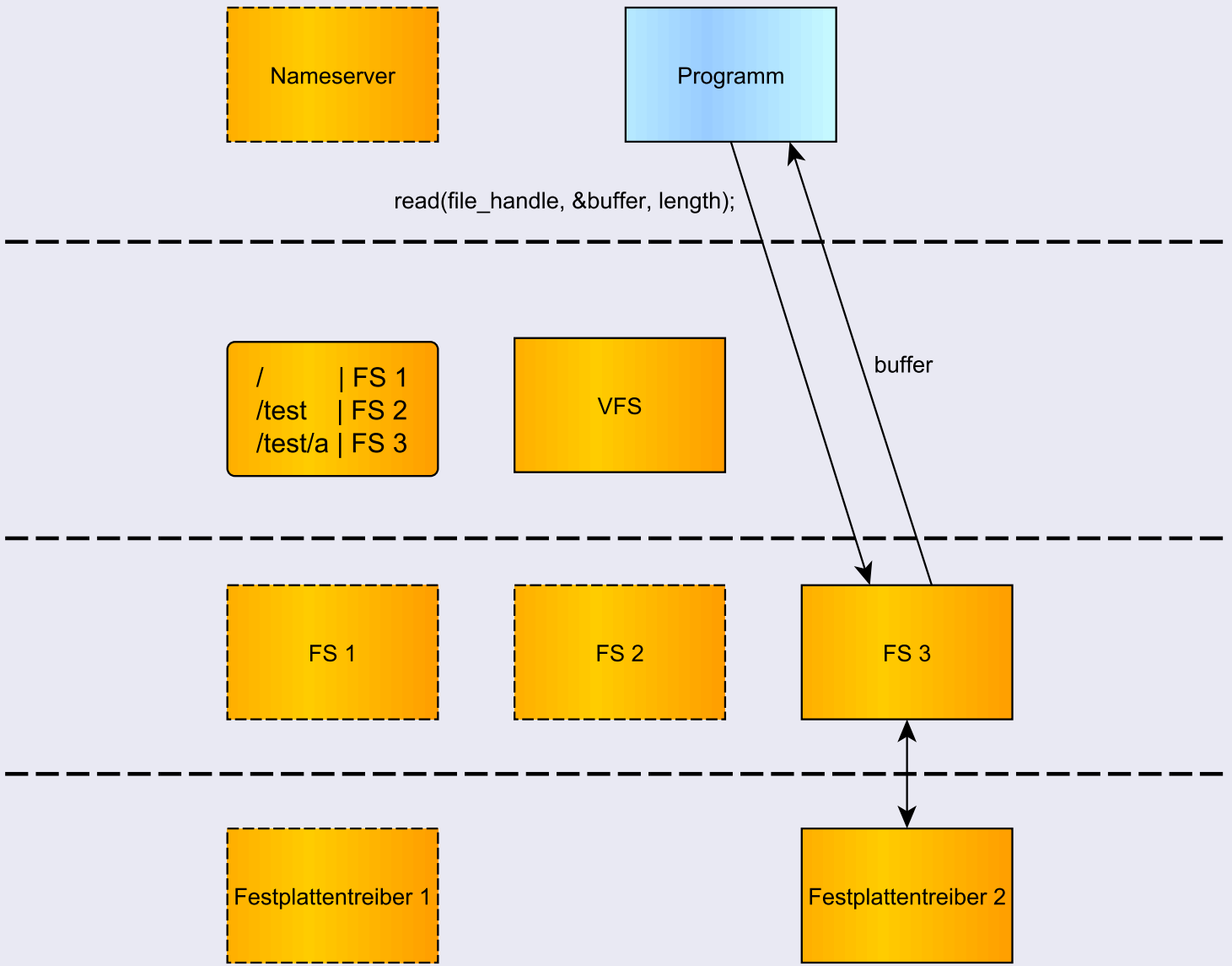
Gemeinsam genutzte Offene Dateien Tabelle

Handle	FS Handle	Flags	Position	Thread ID
0xAB	0xFB98	Read and Write	0	0xC0FFEE

Server Kommunikation - Öffnen von Dateien (Aufruftyp 1)



Server Kommunikation - Lesen von Daten (Aufruftyp 2)



Server Kommunikation - Lesen von Daten (Aufruftyp 2)

Gemeinsam genutzte Offene Dateien Tabelle

Handle	FS Handle	Flags	Position	Thread ID
0xAB	0xFB98	Read and Write	prev. pos + length	0xC0FFEE

Server Kommunikation - Aufruftypen

Welcher Aufruf wird wann verwendet?

Aktion	**Aufruf**
open	Typ 1 (indirekt, VFS)
close	Typ 2 (direkt, FS)
read	Typ 2 (direkt, FS)
write	Typ 2 (direkt, FS)
seek	Typ 1 (indirekt, VFS)
rename	Typ 1 (indirekt, VFS)
delete	Typ 1 (indirekt, VFS)

Die Funktionen *create* und *append* werden durch *open* simuliert.

IDL4 Interfaces - Typen

Exceptions

- exception file_not_found;
- exception dir_not_found;
- exception invalid_call;

Typdefinitionen

- typedef long filehandle;
- typedef long dirhandle;
- typedef char flags;
- typedef long thread_id;
- typedef long offset_t;
- typedef struct stat stat_t;
- typedef struct dirent dirent_t;

IDL4 Interfaces - Dateioperationen

Öffnen / Schließen

- void open(in string pathname, in flags flags, out filehandle filehandle, out thread_id thread_id)
  raises file_not_found, invalid_call;
- void close(in filehandle handle)
  raises invalid_call;

Lesen / Schreiben

- void read(in filehandle handle, inout fpage buffer, in long length)
  raises invalid_call;
- void write(in filehandle handle, in fpage buffer, in long length)
  raises invalid_call;
- void seek(in filehandle handle, inout offset_t offset, in int mode)
  raises invalid_call;

IDL4 Interfaces - Dateioperationen

Dateiverwaltung

- void rename(in string pathname, in string new_name)
  raises file_not_found, invalid_call;
- void delete(in string pathname)
  raises file_not_found, invalid_call;
- void stat(in filehandle handle, inout stat_t stat)
  raises invalid_call;

IDL4 Interfaces - Verzeichnisoperationen

Verzeichnisverwaltung

- void opendir(in string pathname, out dirhandle handle)
  raises dir_not_found, invalid_call;
- void readdir(in filehandle handle, inout dirent_t dirent)
  raises invalid_call;
- void closedir(in dirhandle handle)
  raises invalid_call;

- void mkdir(in string pathname)
  raises invalid_call;
- void rmdir(in string pathname)
  raises dir_not_found, invalid_call;
- void renamedir(in string pathname, in string new_name)
  raises dir_not_found, invalid_call;

IDL4 Interfaces - Verzeichnisoperationen & Sonstiges

Linkverwaltung

- void softlink(in string source_pathname, in string target_pathname)
  raises file_not_found, invalid_call;
- void hardlink(in string source_pathname, in string target_pathname)
  raises file_not_found, invalid_call;
- void unlink(in string pathname)
  raises file_not_found, invalid_call;

Sonstiges

- void closeAllFiles(in thread_id thread_id)
  raises invalid_call;

Zusätzlich benötigte Server:

- Taskserver

Erweiterungsmöglichkeiten - Parallele Dateizugriffe

Threads in FS erlauben parallele Dateizugriffe

Problem: Lese- und Schreibsemantik

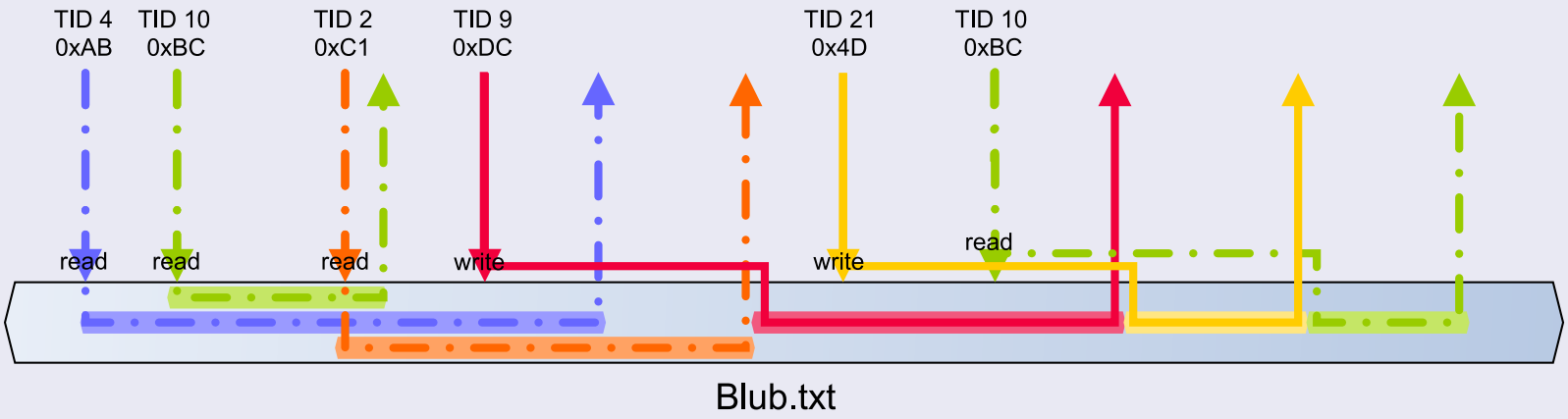
Gleichzeitiges Lesen und Schreiben von Daten innerhalb einer Datei

Lösung: Gegenseitiger Ausschluss von Lese- und Schreibzugriffen

Blockieren von Lesezugriffen bei laufendem Schreibzugriff
Blockieren von Schreibzugriffen bei laufenden Lesezugriffen

Erlaubt sind mehrere gleichzeitige Lesezugriffe oder ein Schreibzugriff.

Erweiterungsmöglichkeiten - Parallele Dateizugriffe



	Aktion	**Mutex Write**	**Atomare Variable**
Lesen	... Lesen ...	Lock Unlock	++ - -
Schreiben	while (variable $>$ 0); ... Schreiben ...	Lock Unlock	

Erweiterungsmöglichkeiten

Vermeidung von Kopiervorgängen

mmap - Einblenden von Dateien in Addressraum

Rechteverwaltung

Zugriffsrechte:

User/Group/Other Read/Write/Execute

Zusätzlich benötigte Server:

- Benutzerverwaltung
- Gruppenverwaltung

Fragen & Kommentare?