

Praktikum angewandte Systemsoftwaretechnik (PASST)

Blockpraktikum

05. Februar 2020

Tobias Langer, Michael Eischer
und Florian Schmaus

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg



Lehrstuhl für Verteilte Systeme
und Betriebssysteme



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

- Projektwahl und Gruppenbildung: 2er Gruppen
- Projektvorstellung
 - 10 min. Präsentation im Plenum + 10 min. Diskussion
 - Vorstellung, Ansatz, erwartete Ergebnisse, Zeitplanung
- 2 Wochen Vollzeit
 - Bei Bedarf tägliches Jour Fixe
 - Zwischentreffen
- Abschlusspräsentation
 - 20 min. Präsentation im Plenum + 10 min. Diskussion
 - Ergebnisse, Erfahrungen, Fazit
- Termin: **02.03.2020 – 13.03.2020**

Zielsetzung (1/2)

Erfolg im Praktikum wird am Erreichen der Zielsetzungen gemessen:

- Gelerntes anwenden
- Selbständige Projektdurchführung im Team
- Entwicklungsprozesse in F(L)OSS-Projekten praktisch anwenden
 - Verwenden entsprechender Werkzeuge (git, Patche, ...)
 - Einbinden der Entwicklergemeinschaft (Features an Upstream)

Ziel

Benutzbare Software für euch, uns und den Rest der Welt

Bewertet wird:

- Lösungsfindung und Umsetzung der Lösung
- Kollaboration zwischen euch
- (Kommunikation und Zusammenarbeit mit Upstream)
- Projekt wird veröffentlicht (Publish or it didn't happen!)
 - Entsprechend: Repo mit Doku, Lizenz, ...

Notenfindung (Wiederholung)

Teilnote	A1	A2	A3	A4	A5	A6	Blockpraktikum
Gewichtung	1	1	2	2	2	2	15

- Semesterbegleitender Teil macht 40% der Punkte aus
- erreichbare Punktezahlen und damit Gewichtung entsprechend dem Umfang der Aufgaben
- Blockpraktikum umfasst die restlichen 60%

StackViz

Dynamische Visualisierung des Callstacks für den Lehrbetrieb

- Visualisierung zur Laufzeit
 - Aufbauend auf gdb/libunwind
- Benutzeroberfläche
 - Hervorhebung von Stackframes, Rücksprungadressen, Verweisen, ...
 - Interpretation von Datenwerten
 - Export als Raster-/Vektorgrafik
- Modularer Aufbau
 - Unterstützung verschiedener Architekturen



Tobias

Dynamischer ELF Loader

ELF Unterstützung für OctoPOS

- Forschungsbetriebssystem
 - Aktuell umgesetzt als Bibliotheksbetriebssystem
 - ... allerdings lange Umlaufzeiten
- Entwicklung eines dyn. Laders
 - Parsen & Interpretieren des ELF Formats
 - Mapping der Sections
 - Auflösen von Symbolen der Systemschnittstelle
- Erste Schritte hin zu Relocation Loader



Tobias

SMIME Unterstützung für mu4e

Emailverschlüsselung für das beste Betriebssystem mit dem schlechtesten Editor

- Email-Client für Emacs
 - PGP/MIME Verschlüsselung, Entschlüsselung, Signatur & Verifikation funktionstüchtig
 - SMIME Verschlüsselung, Entschlüsselung
- Problem: Wie SMIME verschlüsselte Mails lesen?
 - ~> SMIME Entschlüsselung und Verifikation implementieren
 - ... in elisp



Tobias

Dateisystemsandboxing

Zugriffskontrolle für Verzeichnisse mit Overlay

Dateisystemen

- Dynamische Zugriffskontrolle mittels Kernelmodul
 - Rechtekontrolle zur Laufzeit anpassbar
- Aufgabe
 - Einarbeiten in bestehende Overlay Dateisysteme
 - Eigenes Overlaysystem, Rechteverwaltung
 - Entwicklung einer Verwaltungsschnittstelle (sysfs, o.ä.)
 - ggf. Anwendungsbibliothek



Max

Softwaremaßschneiderung

Maßschneiderung von **Shared Libraries** in Snapcraft-Packages

- Snapcraft-Pakete: Anwendung + Abhängigkeiten
 - Vorteil: Keine Dependency Probleme zwischen versch. Anwendungen
 - Nachteil: Overheads durch ungenutzte Funktionalität
- Framework zur Maßschneiderung vorhanden
 - Anwendung auf Snapcraft-Pakete
 - Erweiterung und Anpassung wo notwendig
 - Analyse des erreichbaren Grads der Maßschneiderung



Andreas

Auswirkungen von Intel/AMD Mitigations

Vermessen von Mitigations im Kernel

- Sicherheitsprobleme in Intel/AMD
 - Linuxkernel setzt verschiedene Mitigations um
- Aber:
 - Performanceimplikationen
 - Energieverbrauch
- Praktikum
 - Vermessen der Auswirkungen der Mitigations
 - Korrelation zwischen Performance/Energieverbrauch?



Stefan

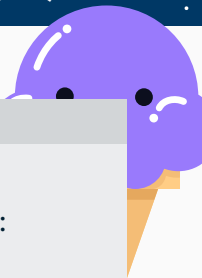
StubsPi - MPStuBS für Raspberry Pi

Raspberry Pi Port auf richtiger Hardware

- Betriebssystemkern für Lehrzwecke
 - Portierung auf Raspberry Pi abgeschlossen
 - Unter Qemu funktionstüchtig
 - ... auf echter Hardware noch nicht



Max



Platin: nicht alles, was glänzt, ist ein Edelmetall

- Programm zur statischen Laufzeitanalyse (**Ruby**)
- Nutzung einiger „spannender“ Sprachkonstrukte:
 - Ententypisierung (engl. duck typing)
 - Spezialisierte Listenimplementierung mittels `module_eval`

- **Aber:** statische Typisierung ist sehr sinnvoll:
Fehlersuche, Codevervollständigung, Typprüfung

~> Sorbet: ein gradueller Typprüfer für Ruby

- **Vorkenntnisse:** Rubykenntnisse sind zwingend erforderlich!



Die schöne Welt des XMPP

- Offenes Protokoll zur Nachrichtenübermittlung
 - Mit ganz vielen Erweiterungen (XEPs):
 - Ende-zu-Ende Verschlüsselung: OMEMO und OpenPGP
 - Multi-User Chats
 - Synchronisierter Chatverlauf:
Message-Archive-Management
 - ... was fehlt ist ein guter TUI-Client, der alle unterstützt!
- ~> Synergieeffekte: Existierende Implementierung nutzen
- Konfigurierbarer Chatclient
 - Typsichere XMPP-Bibliothek (Java)
- ~> Polyglotte Programmierung: Hochsprache und maschinennahe Sprache vereint

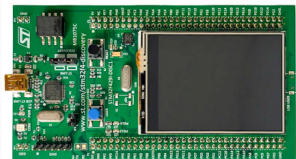


Zeitgesteuertes Betriebssystem ...

- feste Ablaufabelle
- periodische Abarbeitung
- studentische Arbeit

... mit Problemen

- Todeslinienabfrage
- überlappende Einplanung von Aufgaben
- Übernahmeprüfung nichtperiodischer Last



DIY-Oszilloskop: Für jeden was dabei!

- Hardware: Vorverarbeitungsplatine, Signalgenerator
 - Löten, eigener Hardwareentwurf
- eingebettete Entwicklung: Plotten, FFT, Pan & Zoom
 - hardwarenahe Programmierung
- Betriebssystementwicklung: Kommunikation mit Host
 - Treiberentwicklung, Bibliothekseinbindung
- Echtzeiteigenschaften: zeitliche Analyse
 - Messung, Analysetools

Eigene Ideen und Vorschläge

- StackViz
- Dynamischer ELF Loader
- SMIME für mu4e
- Dateisystemsandboxing
- Softwaremaßschneiderung
- Intel/AMD Mitigations
- StubsPI
- Gradual Typing
- TheTruth
- tt-eCos
- DIY-Oszi

Fragen?