Hans-Sachs-Gymnasium Nürnberg
Studienseminar Februar 2023/2025

Schriftliche Hausarbeit

zur zweiten Staatsprüfung
für das Lehramt an Gymnasien

im Fach
Informatik

Thema:

Entwicklung eines Werkzeuges zur Unterstützung der Umsetzung der Inhalte des Lernbereichs Betriebssysteme im Unterricht der Jahrgangsstufe 12

Dise Arbeit © 2024 von Florian Fischer ist lizensiert unter Creative Commons Attribution-ShareAlike 4.0 International CC BY SA.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Seit der Erfindung des Computers haben sich eine Vielzahl von Formen und Anwendungszwecken entwickelt. Seit den Anfängen bis heute gilt, um den Computer als Universalwerkzeug zu verwenden, braucht es etwas oder jemanden, der den Betrieb verwaltet. Es muss zwischen den auszuführenden Programmen umgeschalten und diesen in möglichst einfacher und sicherer Form die verfügbaren Hardwareressourcen zur Verfügung gestellt werden. Diese Aufgabe übernahm bald eine spezielle Softwarekomponente, das Betriebssystem.

Betriebssysteme sind in unserem digitalen Alltag allgegenwärtig. Dennoch ist dieser Fakt oft den Anwenderinnen nicht bewusst, was dafür spricht, dass moderne Betriebssysteme ihrer Aufgabe nachkommen, die einfache und sichere Benutzung der unterschiedlichsten Geräte von Smartwatches, über Mobiltelefone zu klassischen PCs, Rechenzentren und Supercomputern reibungslos zu ermöglichen.

Deren Position zwischen der Hardware und den eigentlichen Anwendungsprogrammen, die dem Computer erst seine Nützlichkeit entlocken, macht sie für viele Personengruppen spannend. Einerseits sind Betriebssysteme trotz ihres Alters stets Bestandteil aktueller Forschung und Entwicklung. Andererseits sind sie Ziel von böswilligen Akteuren, da ein Angriff auf ein Computersystem sich meist zwangsläufig mit dem Betriebssystem beschäftigen muss. Dieses Spannungsfeld macht ihre Behandlung auch bereits im schulischen Informatikunterricht interessant. Ihre Betrachtung erlaubt, viel über allgemein gültige Prinzipien der Informatik zu erfahren, und ermöglicht einen tiefen und spannenden Einblick in die sonst geheimnisvollen und verborgenen Vorgänge unserer täglichen Computernutzung.

In dieser Arbeit wird das Werkzeug School Operating System (SOS), das diese Betrachtung im Informatikunterricht unterstützen soll, vorgestellt und analysiert.

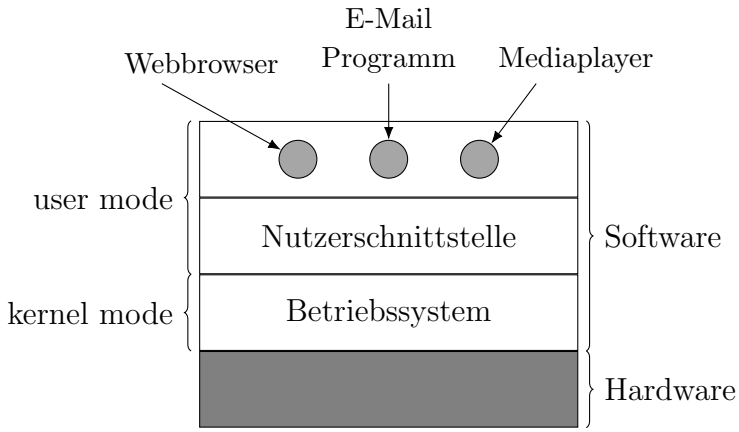


Abbildung 1: Das Betriebssystem zwischen Hardware und Anwendungsprogrammen im Schichtmodell [17, S. 3]

2 Theoretische Grundlagen

Das Betriebssystem ist im vereinfachten Sinn die Software, die privilegiert von der Hardware ausgeführt wird, siehe Abbildung 1. Aber auch unprivilegierte Komponenten, die essenzielle Dienste für die Anwendungsprogramme anbieten, können als Teil des Betriebssystems betrachtet werden. Grundsätzlich hat das Betriebssystem die Aufgabe, die bestehende Hardware den Nutzerprogrammen in geeigneter Weise zur Verfügung zu stellen und deren sichere Verwendung zu gewährleisten [17, S. 2f].

Um die Sicherheit der Ressourcen (englisch *security*) gegen bösartige Dritte, andere Nutzer auf dem System, Schadsoftware, Angriffe über angeschlossene Peripherie-Geräte, zumeist das Netzwerk, sowie, die Betriebssicherheit (englisch *safety*), wie beispielsweise den Verlust von Daten oder das Abstürzen des Systems bei einem Fehler zu verhindern, zu gewährleisten, kooperiert das Betriebssystem mit der zugrundeliegenden Hardware.

Diese bietet zurückgehend auf das Betriebssystem Multics [14] mehrere Schutzdomänen, sogenannte *Ringe*, an. Die Ringe legen fest, welche Fähigkeiten der Hardware von der ausgeführten Software verwendet werden dürfen. Obwohl die für Desktop-PCs weiterverbreitete Prozessorarchitektur x86 vier Ringe anbietet, vergleiche Abbildung 2, verwenden die Betriebssysteme, Linux, MacOS und Windows jedoch nur zwei Ringe, Ring 0 für den *Kernelmodus* und Ring 3 für den *Benutzermodus*. Diese Einschränkung ermöglicht es, einfacher auch Prozessorarchitekturen mit nur zwei Ringen zu unterstützen.

Diese Privilegientrennung erlaubt es dem Betriebssystem, das mit maximalen Berechtigungen in Ring 0 ausgeführt wird, andere Schutzmechanismen der Hardware für die Ausführung der Nutzerprogramme vorzubereiten und zu verwalten.

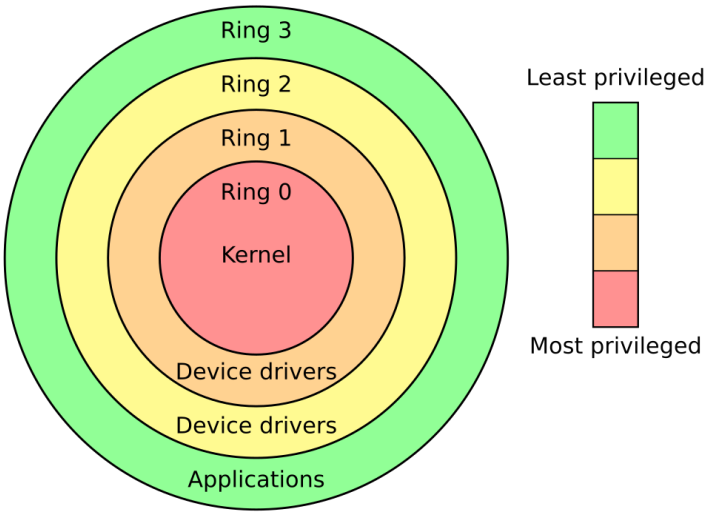


Abbildung 2: Privilegienstufen des protected modes der x86 Prozessorarchitektur [5]

Zum Beispiel wird der physisch vorhandene Speicher in einen zusammenhängenden virtuellen Speicherbereich pro Prozess abstrahiert. Dabei wird der vorhandene Speicher in Abschnitte flexibler Größe, sogenannte Segmente, oder in kleinere Bereiche fester Größe, sogenannte Seiten, aufgeteilt [4]. Die Kombination beider Verfahren ist die vorherrschende Art der Speicherverwaltung. Die Übersetzung von virtuellen Adressen in Physische übernimmt der Prozessor mit Hilfe von durch das Betriebssystem aufgebauten Datenstrukturen. Diese virtuelle Trennung des Speichers sorgt dafür, dass Prozesse nur ihren exklusiven Speicherbereich verwenden können. Dabei wird der Adressraum unterschiedlicher Programme voneinander, sowie der Adressraum von Programmen von dem des Betriebssystems abgetrennt. Allerdings erlauben es Segmente und Seiten gegebenenfalls in mehreren Adressräumen enthalten zu sein und erlauben so Informationen zu teilen [2]. Die Memory Managment Unit (MMU), eine Hardware-Komponente, kontrolliert alle Speicherzugriffe.

Aufgrund des ursprünglichen Ringmodells lässt sich ein Betriebssystem funktional in Schalen, die den entsprechenden Ringen zuordnet werden, strukturieren. Da aber moderne Betriebssysteme meist nur zwei Privilegienstufen verwenden, ist ein Schichtmodell, wie in Abbildung 1 dargestellt, potenziell geeigneter die Komponenten eines Betriebssystems zu veranschaulichen, da dort auch die Hardwareschicht und zusätzliche Schichten, die sich auf derselben Privilegienstufe befinden aber logisch aufeinander aufbauen, verdeutlichen lassen.

Herkömmlich muss ein Anwendungsprogramm, wenn es einen Dienst des Betriebssystems nutzen möchte, um beispielsweise Daten über das Netzwerk zu verschicken, einen Privilegienwechsel in Form eines Systemaufrufs (englisch system call)

vollziehen, welcher mit Kosten verbunden ist. Es ist aktueller Gegenstand der Forschung [16] und mit der aus Linux stammenden Systemschnittstelle `io_uring` [1] bereits in Linux und Windows umgesetzt, Dienste des Betriebssystems statt über einen Wechsel des Rings über Nachrichtenaustausch in geteilten Speicherbereichen zu verwenden.

Eines der fundamentalsten Abstraktionskonzepte von Betriebssystemen ist der *Prozess*. Ein Prozess repräsentiert ein laufendes Programm. Diese Abstraktion erlaubt es, mehrere Programme scheinbar gleichzeitig auf nur einem Rechenkern auszuführen.

Ein Prozess besteht aus allen Datenstrukturen, die zur Programmausführung benötigt werden, beispielsweise einem exklusiven Speicherbereich, sowie dem auszuführenden Programmcode. Jeder Prozess hat einen *Zustand*, anhand dessen er in der Ablaufplanung des Betriebssystems, berücksichtigt wird. Wartet ein Prozess auf eine Ressource, Daten von der Festplatte oder Nutzereingaben, wird er von der Ablaufplanung ausgenommen bis die Ressource verfügbar ist.

Um die Sicherheit des Systems zu gewährleisten, hat jeder Prozess zugewiesene Berechtigungen, oft sind diese mit dem *Besitzer* und der Gruppe des Prozesses verbunden. Allerdings existieren auch Berechtigungskonzepte basierend auf einzelnen „Befähigungen“ (englisch *capability*), die eine präzisere Verwaltung von Berechtigungen erlauben. Einige Eigenschaften, allen voran die Berechtigungen eines Prozesses aber auch der Besitzer, werden beim Erstellen des Prozesses vom *Elternprozess* geerbt. Da ein Prozess immer nur von einem anderen Prozess erzeugt werden kann, bilden alle Prozesse des Systems einen Baum. Wobei die Wurzel des Baums vom Betriebssystem gestartet wird.

Die hierarchische Gliederung in eine aus dem potentiell vorangegangenen Lernbereich 3 [8] bekannte Baumstruktur ist eine fundamentale Idee der Informatik [15, S. 92ff] und spielt eine herausragende Rolle in SOS bei der Analyse der Komponenten eines Betriebssystems.

Eine weitere essenzielle Abstraktion von Betriebssystemen ist die *Datei*, um die Eigenheiten verschiedener Hardware zu verstecken. Die Abstraktion *Datei* kann nicht nur für persistente Speicherbereiche verwendet werden, sondern auch um Dateien zu strukturieren, werden besondere Dateien, sogenannte Verzeichnisse (englisch *directory*), verwendet. Diese erlauben es, Dateien in einem Dateisystem in einer Baumstruktur zu gruppieren (vergleiche Abbildung 3).

Dateien als uninterpretierte Folge von Bytes mit bekannter *Länge* zu behandeln hat sich gegen strukturiertere Formen von Dateien durchgesetzt [17, S. 267].

Neben regulären Dateien und Verzeichnissen werden auch Hardwaregeräte in

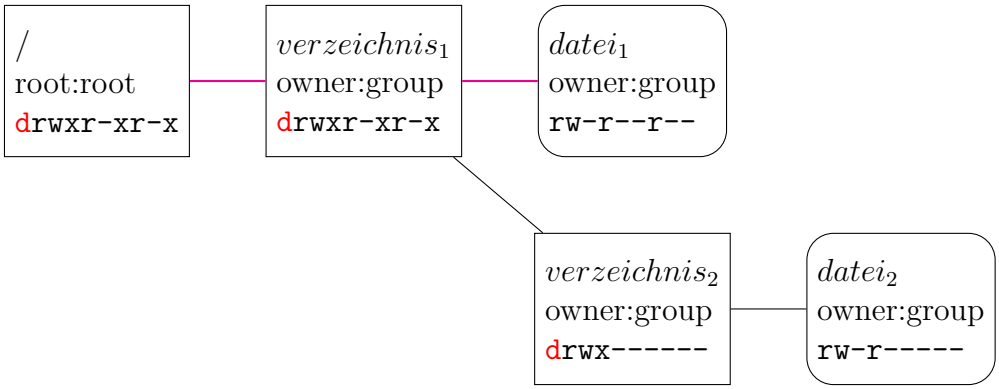


Abbildung 3: Baumdarstellung eines exemplarischen Dateisystems beginnend mit dem Wurzelverzeichnis `/`. Der absolute Pfad der Datei $datei_1$ lautet `/`$verzeichnis_1$`/`$datei_1$.

UNIX-ähnlichen Systemen als Dateien bereitgestellt. Das hat den Vorteil, dass die bestehenden Zugriffs- und Kontrollmechanismen der Dateiverwaltung auch auf Hardwaregeräte, wie USB-Sticks, angewendet werden können. Außerdem können Programme, die zum Umgang mit Dateien entwickelt wurden, so auch für Hardware benutzt werden.

Da in einem System mehrere Dateien mit dem gleichen Namen existieren können, wird um eine bestimmte Datei verwenden zu können, nicht nur ihr Name, sondern auch ihr Ort im Dateisystem benötigt.
Der Ort einer Datei kann auf zwei Arten angeben werden als **absoluter** oder **relativer** Pfad. Ein Pfad besteht aus einer beliebigen Liste an Verzeichnissen und einem optionalen abschließenden Dateinamen getrennt von *Pfadseparatoren*. Absolute Pfade beginnen immer mit dem Wurzelverzeichnis des Dateisystems. Relative Pfade werden ausgehend vom aktuellen Arbeitsverzeichnis des Prozesses interpretiert. Um die beiden Sicherheits-Schutzziele *security* und *safety* zu gewährleisten, erzwingt das Betriebssystem, dass der verwendende Prozess die entsprechenden Berechtigungen besitzt.

Eine sehr rudimentäre, aber dennoch in UNIX-ähnlichen Systemen weit verbreitete Art, die Berechtigungen einer Datei zu verwalten, ist diese in zwölf Bits zu speichern. Neun dieser Bits werden, wie in Abbildung 4 dargestellt, in drei Klassen á drei Bits aufgeteilt. Diese Klassen stehen für den Eigentümer, die Benutzergruppe der Datei, sowie alle anderen. Die drei Bits `rwx` (**r**ead **w**rite e**x**ute) einer Klasse geben an, ob diese die Datei lesen, schreiben, ausführen oder falls es sich um ein Verzeichnis handelt, auf andere Dateien innerhalb des Verzeichnisses zugreifen darf. Vor der Verwendung einer Datei wird zunächst die Klasse des Prozesses, der die Datei verwenden möchte, bestimmt und anhand derer geprüft, ob die Operation

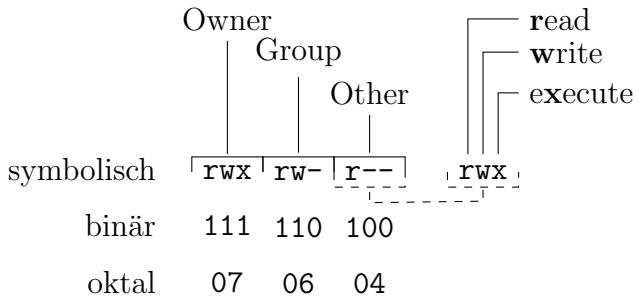


Abbildung 4: Neun Berechtigungsbits, die die erlaubten Arten des Zugriffs auf eine Datei für die drei Klassen Besitzer, Benutzergruppe und Weitere regeln

erlaubt ist.

Neben den neun grundlegenden Bits für die drei Klassen existieren noch drei zusätzliche Bits für erweiterte Dateiberechtigungen. Eines dieser Bits wird „*SUID*“ (**S**et **UID**) genannt. Ist dieses Bit gesetzt, wird das Programm mit dem Eigentümer der Programmdatei statt dem Benutzer des Elternprozesses ausgeführt. Dies wird verwendet, um beispielsweise Nutzern eine kontrollierte Ausweitung ihrer Berechtigungen zu ermöglichen.

Moderne Dateisysteme unterstützen darüber hinaus, eine spezifischere Steuerung der Berechtigungen auf Basis einer *Zugriffssteuerungsliste*.

Die Dateiberechtigungen UNIX-ähnlicher Dateisysteme sind ein geeignetes Beispiel Kontrollmechanismen eines Betriebssystems für den schulischen Kontext aufgrund ihrer Einfachheit und der Tatsache, dass sie darüber hinaus gut ohne Systemprogrammierung erfahrbar sind.

3 Didaktische Analyse

3.1 Einordnung in den Lehrplan

Der ab dem Schuljahr 2024/2025 gültige LehrplanPlus der 12. Jahrgangsstufe, enthält im Lernbereich 5 des erhöhten Anforderungsniveaus den Themenkomplex *Betriebssysteme* [8]. Im Folgenden werden die geforderten Kompetenzerwartungen und ihre Behandlung in SOS beleuchtet.

> Die Schülerinnen und Schüler erläutern den Zweck sowie die Aufgaben eines Betriebssystems und beschreiben dessen prinzipiellen Aufbau anhand eines Schalenmodells. (LehrplanPlus [8])

Die Aufgaben eines Betriebssystems werden knapp im Theorieteil des SOS Arbeitsheftes behandelt (siehe Anhang Seite A-9). Wobei die Interaktion mit der Hardware und potenziell benötigte Privilegienwechsel bei der Verwendung von Diensten des Betriebssystems nicht im Rahmen von SOS behandelt werden. Außerdem wird die Rolle des Betriebssystems als Schichtmodell statt als Schalenmodell veranschaulicht, aufgrund der reelleren Darstellung, wie in Abschnitt 2 auf Seite 3 beschrieben.

> [...] vergleichen die Anforderungen, die an Betriebssysteme in unterschiedlichen Einsatzszenarien (z. B. Einzel- und Mehrbenutzerbetrieb) gestellt werden. (LehrplanPlus [8])

Die Anforderungen eines Einzelbenutzerbetriebes werden nicht thematisiert, wohingegen die Anforderungen an ein Betriebssystem bei einem Mehrbenutzerbetrieb in SOS besonders in Aufgabe Geheimnisse aktiv erfahrbar sind. Die Bedeutung von Dateiberechtigungen werden immer im Kontext mehrerer Nutzer interpretiert.

> [...] nutzen und beschreiben Funktionen und Mechanismen von Betriebssystemen, die dem Schutz und der Sicherheit von elektronischen Geräten dienen, und erläutern, welche Sicherheitsrisiken mit Betriebssystemen verbunden sind. Dabei betrachten sie insbesondere die Rechteverwaltung in Betriebssystemen. (LehrplanPlus [8])

Dateiberechtigungen sind der Hauptfokus von SOS als eine einfache und zugängliche Form der Rechteverwaltung. An mehreren Stellen in SOS und im Arbeitsheft wird auf potenzielle Risiken mangelnder Rechteverwaltung hingewiesen. Auch der Schutz von Geräten, die in SOS und in UNIX-ähnlichen Systemen als Dateien im Dateisystem abgebildet sind, durch die gleichen Mechanismen wie für Dateien wird im Arbeitsheft erwähnt.

> [...] beschreiben und bewerten unterschiedliche Scheduling-Strategien, die von Betriebssystemen bei der Prozessverwaltung eingesetzt werden, um den zeitlichen Ablauf von Prozessen zu regeln. (LehrplanPlus [8])

Die Betrachtung unterschiedlicher Scheduling-Strategien ist tendenziell im Rahmen von SOS denkbar, siehe Abschnitt 4, aber nicht Bestandteil dieser Arbeit.

Folgende vom Lehrplan geforderte Inhalte zu den Kompetenzen werden in SOS betrachtet [8]:

> zentrale Aufgaben eines Betriebssystems: Prozessverwaltung, Speicherverwaltung, Geräte- und Dateiverwaltung, Benutzerverwaltung (LehrplanPlus [8])

Alle diese Begriffe werden im Arbeitsheft erklärt und behandelt, aber ausschließlich die Dateiverwaltung wird dabei vertieft.

> Betriebssystemkern (Kernel), Kernelmodus (Kernel-Mode), Benutzermodus (User-Mode), Dienst, Schnittstelle, Systemaufruf (LehrplanPlus [8])

Die Begriffe *Kernel*, *Benutzer-*, sowie *Kernelmodus* werden im Theorieteil (siehe Anhang Seite A-9 des Arbeitsheftes erklärt. Die Tatsache, dass die Trennung in Benutzer- und Kernelmodus bei der Verwendung eines Dienstes gewöhnlich einen durch die Hardware unterstützten Privilegienwechsel in Form eines Systemaufrufs benötigt, wird wie bereits erwähnt in SOS nicht behandelt. Da SOS Fokus auf die praktische Erfahrbarkeit legt und Systemaufrufe praktisch hauptsächlich durch Systemprogrammierung erfahrbar sind, ist deren rein theoretische Betrachtung nicht Teil von SOS.

> Zugriffsrechte, Zugriffskontrolle, Authentifizierung (LehrplanPlus [8])

Diese Begriffe tauchen nicht explizit in SOS auf, deren Betrachtung ist aber potenziell möglich und findet implizit in den Aufgaben des Arbeitshefts statt.

Der ebenfalls in Lernbereich 5 enthaltene Themenkomplex der Nebenläufigkeit [8] ist generell mit den in der Schule erworbenen Kompetenzen einfacher in einer höheren Programmiersprache erfahrbar. Die Thematik könnte zwar durchaus im Rahmen eines Betriebssystems in Anknüpfung an Scheduling behandelt werden, aber die Analyse und die Umsetzung sind im Rahmen des Betriebssystemkontexts komplexer.

Generell ist die Menge der im Theorieteil des Arbeitshefts enthaltenen Informationen, sowie die Schwierigkeitsgrade der Übungsaufgaben an der Zielgruppe des erhöhten Anforderungsniveaus orientiert. Die Einstiegsaufgaben (vergleiche Anhang Seite A-10) starten gezielt einfach und enthalten viel Hilfestellung, um Überforderung zu vermeiden, wohingegen insbesondere der letzte Teil der letzten Aufgabe Geheimnisse schwierig ist und Interessierte auch fordern soll.

3.2 Mögliche Unterrichtssequenz

Der ebenfalls im grundlegenden Anforderungsniveau enthaltene Lernbereich 5 behandelt mit neun Stunden das Themenfeld der Nebenläufigkeit [9]. Im erhöhten Anforderungsbereich umfasst dieser Lernbereich 23 Stunden [8]. Aus der Differenz ergibt sich, dass das Thema Betriebssysteme mit maximal 14 Stunden veranschlagt ist.

Ein potenzieller Unterrichtsverlauf könnte grob, wie in Abbildung 5 dargestellt, strukturiert werden. Die ersten vier Stunden werden genutzt, um die vom Lehrplan geforderten Begriffe, Aufgaben und zugrundeliegende Techniken eines Betriebssystems zu betrachten. Anschließend erfolgt eine Wiederholung sowie praktische Anwendung der Grundlagen in der Auseinandersetzung mit SOS. Zusätzlich wird die Aufgabe eines Betriebssystems, einen sicheren Betrieb zu gewährleisteten, und die dafür eingesetzten Kontrollmechanismen vertieft und praktisch erfahren. Anschließend folgt die Betrachtung unterschiedlicher Scheduling-Strategien. Die verbleibenden 2 Stunden könnten flexibel für einen der drei vorangegangenen Bereiche nach Bedarf verwendet werden.

2 h Theoretische Grundlagen

2 h Technische Grundlagen

4 h Zugriffskontrolle mit SOS

4 h Scheduling

2 h Flexibel

Abbildung 5: Grobe Sequenz eines Unterrichtsverlaufs zum Thema Betriebssysteme

3.3 Das Werkzeug: SOS

Alle Aussagen dieser Arbeit beziehen sich auf die Version 0.2.8 von SOS zu finden unter `https://github.com/fischerling/SOS/releases`.

Bei SOS handelt es sich um eine Kombination aus einer portablen Windows-Version der frei unter GPL-2 lizenzierten virtuellen Maschine QEMU, dem angepassten Lehrbetriebssystem MentOS der Universität Verona, sowie begleitendem Unterrichtsmaterial in Form eines Arbeitsheftes.

Die drei Ziele von SOS sind:

- Der Fokus auf die einfache Handhabe sowohl durch Schüler und Schülerinnen (SuS) in der Anwendung als auch durch Lehrkräfte in der Vorbereitung und Umsetzung ihres Unterrichts
- Das Erfahrbarmachen schwierig greifbarer Aspekte, die tief im Stapel der Technologie im Betriebssystem (vergleiche Abbildung 1) angesiedelt sind

- Das Potenzial, nachhaltiges Interesse durch die leicht zugängliche weiterführende Komplexität, begleitet von umfangreicher Dokumentation, sowie der Lizenzierung als freie Software, zu wecken

Die Einführung in eine tiefere technische Beschäftigung mit Betriebssystemen und die Systemprogrammierung ist mit SOS nicht ohne weiteres möglich, aber ist durch das verwendete Lehrbetriebssystem MentOS leicht zugänglich. Dies ist auch der Grund warum die Kombination von QEMU und MentOS einer beispielsweise auf Linux basierenden Lernumgebung vorgezogen wurde. Bestehende Betriebssysteme wie Linux sind aufgrund ihrer für den Alltagseinsatz benötigten Komplexität nicht von SuS oder Lehrkräften mit angemessenem Aufwand durchdring- oder modifizierbar.

Konzeptionell ist SOS zum Einsatz im Unterricht als Begleitung für Arbeitsphasen in Einzel- oder Partnerarbeit gedacht, um die echte Lernzeit zu erhöhen [12]. Außerdem erscheint eine intensive Interaktion mit SOS nur mit maximal zwei Personen an einem Computerarbeitsplatz realistisch.

Im Gesamtkonzept der informatischen Bildung nach Schubert u. a. [15, S. 56ff] greift SOS besonders auf die im Spiralcurriculum in der Jahrgangsstufen 6 und 7 erworbenen Kompetenzen der Komponenten *a1. Mensch-Maschine-Interaktion* sowie *a3. Wirkprinzipien von Informatiksystemen* zurück. Die im Fachprofil Informatik [6, Abschnitt 3] beschriebene objektorientierte Betrachtung von Software und ihren Komponenten wird in SOS mit den Komponenten eines Betriebssystems fortgeführt.

SOS beruht maßgeblich auf den Komponenten *c.1 Analyse und Spezifikation*, sowie *c3 Erprobung* der Sekundarstufe II [15, S. 56ff], um die geforderten Inhalte im Bereich Betriebssystem zu erarbeiten.

3.4 MentOS

MentOS ist ein freies unter den Bedingungen der MIT-Lizenz veröffentlichtes Lehrbetriebssystem. Dessen Designziele [19] sind soviel Realitätsnähe wie nötig, um einen echten Einblick in Betriebssysteme zu vermitteln, und andererseits soviel Einfachheit wie möglich, um tatsächliches Eingreifen, Erfahren und Durchdringen zu ermöglichen. Es ist vom Aufbau stark an das freie Betriebssystem Linux angelehnt, was dazu führt, dass erworbenes Wissen möglichst leicht in die „echte Welt“ übertragen werden kann.

MentOS ist ein in C geschriebenes 32-Bit single-core Betriebssystem mit Privilegientrennung in Benutzer- und Kernelmodus, Speicher- und Prozessverwaltung, sowie Unterbrechungsbehandlung und Systemaufrufen.

Es ist als Basis für einen universitären Betriebssystem-Kurs entwickelt worden und dazu gedacht von Studierenden angepasst und neu übersetzt zu werden.

In SOS wird MentOS aber tatsächlich ausschließlich aus der *shell* im Benutzermodus verwendet, da Systemprogrammierung in C nicht Teil des Schullehrplans ist. Deshalb mussten einige Verbesserungen am Betriebssystemkern eingepflegt werden, die eine tatsächliche Nutzung des Betriebssystems ermöglichen. Zusätzlich ist im Rahmen dieser Arbeit die Dokumentation sowie die enthaltenen Nutzerprogramme maßgeblich ergänzt worden.

Alle Änderungen, die dazu gedacht sind in MentOS aufgenommen zu werden, sind in englischer Sprache verfasst. Eine zusammenfassende Liste aller im Rahmen dieser Arbeit entstandenen Änderungen an MentOS ist im Anhang auf Seite A-18 enthalten.

Ich habe als Betriebssystem für SOS, MentOS ausgewählt, anstatt eine Lernumgebung auf Basis von Linux umzusetzen, da MentOS durch seinen bereits oben erwähnten vergleichsweise einfachen Aufbau und gut dokumentierten Code, einerseits Interessierten tiefere Einblicke ermöglicht, und andererseits es Lehrkräften erlaubt verhältnismäßig einfach Modifikationen am Betriebssystemkern vorzunehmen.

Zu MentOS existieren Lehrmaterialien [18], die über den Anwendungszweck in deutschen Schulen hinausgehen und das Erlernen von Systemprogrammierung in C begleiten.

3.5 Das `intro` Programm

Das in SOS enthaltene und im Arbeitsheft erwähnte `intro` Programm bildet in Kombination mit der ebenfalls in SOS enthaltenen Dokumentation, weitestgehend in Form von *manual pages*, die Grundlage und Hilfestellung, wie die Aufgaben in dem neuen Problembereich Betriebssysteme allgemein und SOS im Spezifischen bearbeitet werden können [10, siehe *supportive information*].

Die Einführung erfolgt in acht Schritten. Der Hinweistext des aktuellen Schritts wird beim Ausführen des Programms angezeigt. Um zum nächsten Schritt zu gelangen, muss eine Aufgabe gelöst werden. Jeder Schritt enthält Hinweise, die zur Lösung der aktuellen Aufgabe hinführen und mit Fortschritt der Lernenden im `intro` Programm stetig abnehmen [10, siehe *procedural information*]. Die meisten Überleitungsaufgaben enthalten von der gegebenen Antwort abhängige ergänzende Tipps, die beim Lösen der Aufgaben unterstützten. Bei korrekter Antwort enthält die Reaktion weiterführende Informationen oder Hinweise. Im Folgenden wird der Inhalt der Schritte kurz aufgeführt, eine vollständige Auflistung der Texte des `intro` Programms befindet sich im Anhang.

Schritt 0 Erläuterung, wie Programme in der Kommandozeile aufgerufen werden (Anhang Seite A-1)

Schritt 1 Verweis auf weiterführende Informationen in Abschnitt 1.1 des Arbeitshefts; Hinweis, wie die im System enthaltenen Dokumentationsseiten (manual pages) gelesen werden können (Anhang Seite A-1); Dieser Schritt ermöglicht es den SuS selbständig bei Unklarheiten erneut Hilfe zu erhalten und bietet weiterführende Informationen. Die Frage nach der Datei, die beim Start einer interaktiven shell-Sitzung ausgeführt wird, führt an eine der drei Möglichkeiten Aufgabe 2.3 zu lösen heran.

Schritt 2 Erklärung zum Lesen von Verzeichnissen, dem Ursprung des Dateisystems, sowie absoluten Pfaden (Anhang Seite A-2)

Schritt 3 Einführung des *Current Working Directory (CWD)* und Bewegung im Dateisystem mit dem shell-Befehl `cd` (Anhang Seite A-3); Die Schritte 2 und 3 erleichtern die Bearbeitung der Aufgaben 2.1 und 2.2, in denen Dateisysteme untersucht werden sollen.

Schritt 4 Anleitung zum Lesen von Dateien (Anhang Seite A-3); Die Überleitungsfrage thematisiert die Datei `/etc/passwd`, welche die Benutzerverwaltung von UNIX-ähnlichen Betriebssystemen darstellt. Bei korrekter Antwort wird auf zusätzliche Informationen in der Dokumentationsseite zu `passwd` verwiesen.

Schritt 5 Einführung des Kopierens von Dateien und relativen Pfaden (Anhang Seite A-4)

Schritt 6 Erklärung zum Löschen von Dateien; Die Überleitungsfrage wiederholt das Konzept absoluter Pfade (Anhang Seite A-4).

Schritt 7 Anleitung zum Umleiten der Ausgabe eines Prozesses (Anhang Seite A-5); Die letzten drei Schritte bereiten auf die Aufgabe 2.3 vor.

3.6 Das Arbeitsheft

Das in SOS enthaltene Arbeitsheft ist so gestaltet, dass es entweder als Selbstlernkurs oder als Sicherung beziehungsweise Begleitung des Unterrichtsgangs mit SOS eingesetzt werden kann. Es kann selbstverständlich auch als Wissensgrundlage für Lehrkräfte bei der Gestaltung des Unterrichts herangezogen werden.

Es enthält nach dem Vorbild vieler hervorragender Arbeitshefte der Universität Passau, in knapper Form alle geforderten theoretischen Grundlagen, sowie anwendungsbezogene Übungsaufgaben zu den behandelten Lerninhalten des Lernbereichs Betriebssysteme.

Der vorangestellte Theorieblock liefert die Minimalgrundlage für die im LehrplanPlus geforderten Inhalte zu Betriebssystemen (ausgenommen des Schedulings) und bildet die Wissensbasis, um die folgenden Arbeitsaufträge zu bearbeiten.

Bei vorangegangener Einführung der entscheidenden Betriebssystemkonzepte, kann auf das Behandeln des Theorieblocks verzichtet werden, er dient in diesem Fall nur noch als Referenz oder gegebenenfalls Sicherung des Behandelten.

Die im Arbeitsheft enthaltenen Aufgaben mit Ausnahme des `intro` Programms und der Geheimnisse Aufgabe müssen im Unterricht verbessert werden.

3.6.1 Berechtigungs-Bits

Nach der theoretischen Einführung von Dateiberechtigungen in UNIX-ähnlichen Systemen, die durch Abbildung 4 visuell unterstützt wird, folgt die erste Aufgabe (siehe Anhang Seite A-14). Diese knüpft an den Lernbereich 2 *Codierung und Verschlüsselung* der 11. Jahrgangsstufe [7] an. Die erworbene Kompetenz zur Umwandlung von Zahlen unterschiedlicher Stellenwertsysteme wird benutzt, um aus der oktalen Darstellung gängiger Berechtigungen für Verzeichnisse in die Binäre und damit anschließend in die symbolische Darstellungsform der Berechtigungen zu überführen.

Diese Umwandlung dient der Versinnbildlichung der unterschiedlichen Ebenen der Darstellung, von der maschinennahen Darstellung der einzelnen Bits zu der für den Menschen verständlichen semantischen Darstellung mittels den mnemonischen Symbolen `rwx`.

Anschließend soll mit Hilfe des Programms `stat`, die Dateiberechtigungen folgender Dateien untersucht werden, um den ersten Kontakt mit den kontextspezifischen Berechtigungen einzelner Dateien herzustellen. Alle Dateien mit Ausnahme der Datei `/home/bob` haben UID und Gruppenkennung (GID) 0.

`/bin/ls`	`r-xr-xr-x`	Von jedem les- und ausführbare Datei, da Programme im System mit mehreren Benutzern geteilt werden
`/bin/doas`	`r-sr-xr-x`	Das gesetzte *SUID* Bit erlaubt es berechtigten Nutzern, Programme mit UID 0 auszuführen. Bei `doas` handelt es sich um eine wie in allen Desktop-Betriebssystemen enthaltene kontrollierte Form der Rechteausweitung.
`/etc/passwd`	`rw-r--r--`	Von jedem lesbar und nur von *root* änderbar, um Informationen über die Nutzer des Systems bereit zustellen, aber diese vor Veränderung zu schützen
`/etc/shadow`	`rw-------`	Nur von *root* lesbar damit die enthaltenen Passwörter nicht einsehbar sind
`/home/bob`	`rwx------`	Nur vom Eigentümer *bob* benutzbar

3.6.2 Kartografie der Beispiellandschaft

Die Aufgabe Kartografie der Beispiellandschaft (siehe Anhang Seite A-15) erleichtert SuS, die Interpretation der abstrakten und stark formalisierten Dateiberechtigungen, indem diese mit den Eigenschaften alltäglicher Landschaften, Räumen und Gegenständen in Verbindung gebracht werden. Die Verdeutlichung der formalsprachlichen Berechtigungen mit Hilfe einer gegenständlichen Darstellung dient dazu das anschließende Bewerten des status quo eines realen Dateisystems [20], sowie die Beurteilung und das Erkennen von Sicherheitsproblemen in den Aufgaben Kartografie des Dateisystems und Geheimnisse, zu erleichtern [11].

Im Folgenden werden die im Verzeichnis `Landschaft` enthaltenen Dateien und ihre Berechtigungen kurz skizziert.

- WilderWesten - `drwxrwxrwx`: Jeder darf alles.
- Museum - `dr-xr-x-rx`: Jeder darf rein (`x`) und gucken (lesen `r`) aber nicht anfassen (schreiben).
  - Gaestebuch - `rw-rw-rw-`: Jeder darf lesen und schreiben. Vandalismus ist nicht auszuschließen.
  - Schaufel - `r-xr-xr-x`: Für jeden ausführbare Datei
  - Generalschluessel - `r-sr-xr-x`: Für jeden ausführbare Datei, die eine *root* Shell-Sitzung startet und damit uneingeschränkte Rechte gewährt

  - Schaubild - `r--r--r--`: Für alle nur lesbar

- Wohnung - `drwx------`: Nur von Besitzer nutzbar.
- Vereinsheim - `dr-xrwx---`: Kollektiv von einer Gruppe änderbar.
- Briefkasten - `drwx-w--w-`: Alle dürfen neue Dateien erzeugen (`w`) aber nicht mehr lesen, nur der Besitzer, die Post, darf leeren.
- Nebel - `d--x--x--x`: Jeder darf rein aber man muss wissen, wo man hin will.

3.6.3 Kartografie des Dateisystems

Die Aufgabe Kartografie des Dateisystems (siehe Anhang Seite A-16) wendet, das in der vorangegangenen Aufgabe erworbene Gefühl und Verständnis für Dateiberechtigungen und ihre Bedeutung auf ein realitätsnahes Beispiel, das Dateisystem von SOS selbst, an. Die beiden Kartografie-Aufgaben unterscheiden sich maßgeblich in den untersuchten Inhalten, um eine höhere Abstraktion zu ermöglichen [10].

Abbildung 6 zeigt eine mögliche Lösung der Aufgabe. Dem Baumdiagramm ist zu entnehmen, dass alle Dateien, die keine Nutzerdateien sind, dem *root* Nutzer gehören und meist nur von diesem schreibbar sind. Dadurch wird garantiert, dass der zwischen Nutzern geteilte Zustand des Systems, wie beispielsweise die Konfiguration im Verzeichnis `/etc` oder die enthaltenen Programme unter `/bin` vor einer böswilligen oder fehlerhaften Veränderung geschützt sind. Das bedeutet, dass das systemweite Installieren neuer Programme und das Ändern von Systemkonfiguration nur durch privilegierte Benutzer möglich ist. Im Gegensatz dazu sind nahezu alle Dateien des Betriebssystems von allen Nutzern des Systems lesbar.

Alle Dateien, die den Nutzern gehören, wie ihre Dokumente und Bilder, befinden sich in eigenen exklusiven Teilbäumen unter `/home`, den Home-Verzeichnissen der Nutzer und sind nur von diesen zugreifbar. Dies garantiert die Sicherheit der eigenen Dateien vor dem Zugriff Dritter.

3.6.4 Geheimnisse

Die letzte Aufgabe (siehe Anhang Seite A-17) stellt eine kleine Herausforderung dar und ist im Vergleich zu den vorangegangenen explorativen Aufgaben eine deutliche Progression. Das gewählte Setting, die Exfiltration und das Schützen von Information erinnert an Hacking und Capture The Flag Spiele aus IT-Sicherheitswettbewerben. Zusätzlich sorgen einerseits das Erleben der Motivationsgrundbedürfnisse und die dadurch begünstigte intrisische Motivation [3], sowie das Wiederaufrufen und die Anwendung des Gelernten [13] andererseits für einen hohen Lernerfolg.

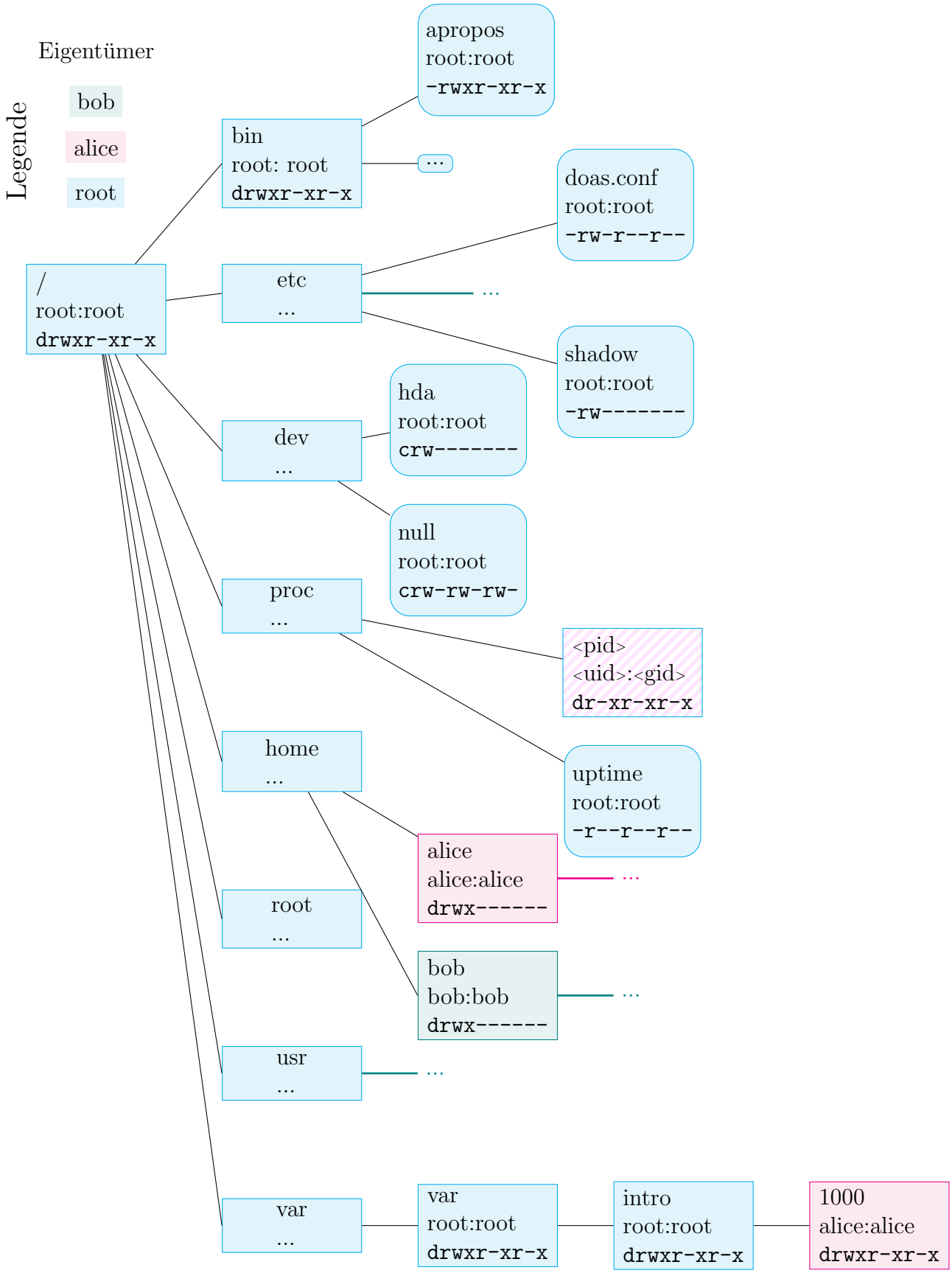


Abbildung 6: Nach Besitzern gefärbtes Baumdiagramm des Dateisystems von SOS; Dicke eingefärbte Kanten symbolisieren, dass beliebige Dateien mit dem selben Eigentümer folgen können. Die Schraffur der Verzeichnisses <pid> symbolisiert, dass der Eigentümer unterschiedlich ist und von der UID des Prozesses mit <pid> abhängt.

Auch die drei Teilaufgaben selbst weisen eine deutliche Progression ihrer Schwierigkeit auf. Im Folgenden werden die drei Teile der Aufgabe knapp vorgestellt:

1. Das Schützen des eigenen Geheimnisses; Hierbei müssen die SuS erst einmal erkennen, dass die Datei *secret.txt* aktuell nicht *alice*, sondern *bob* gehört. Da die Datei aber von Alice lesbar ist (`rw-rw-rw-`), kann die Datei kopiert werden. Anschließend kann das Original gelöscht werden, da es sich in Alice' Home-Verzeichnis befindet, das *alice* verändern (schreiben) darf. Danach muss die Kopie wieder in eine Datei mit Namen *secrets.txt* kopiert werden, und die Standardberechtigungen `rw-------` sorgen dafür, dass nur noch *alice* die Datei lesen darf.

2. Um die Datei *secrets.txt* von Bob zu lesen, reicht die Information, dass sie existiert aus, da die Berechtigungen für Bobs Home-Verzeichnis `rwxrwx-wx` zwar das Auflisten der Dateien verhindert nicht aber deren Benutzung, da das `x` Bit für *others* gesetzt ist. Da Bobs *secrets.txt* Datei ebenfalls von jedem lesbar ist (`rw-rw-rw-`), kann der Inhalt einfach ausgelesen werden.

3. Die mit Abstand schwierigste Aufgabe ist es, die Informationen aus der Datei *top_secret.txt* zu erfahren, da diese Datei nur von Bob lesbar ist. Es existieren, meines Wissens nach drei Möglichkeiten die Informationen zu erhalten.

   (a) Alice ist in der Gruppe *wheel* und darf damit das Programm `doas` verwenden, um Programme als *root* Nutzer, das heißt mit UID 0, auszuführen. Damit kann die Datei ausgelesen werden. Diese Möglichkeit, auf die im Arbeitsheft durch die Erwähnung von `doas` hingewiesen wurde, soll dafür sensibilisieren, dass *root* oder *Admin* Nutzer auf einem System potenziell alles machen dürfen. Wenn also auf einem System eine dritte Person Zugriff auf diese Privilegien hat, muss auch dieser dritten Person vertraut werden. Dies ist insbesondere wichtig für den Einzug von Cloud-Computing in unseren Alltag. Alle Daten, die in einer Cloud gespeichert werden, sind, soweit nicht vor dem Hochladen verschlüsselt wurde, nicht vertraulich.

   (b) Die zweite Möglichkeit ist die Information nicht aus der Datei, sondern aus dem Quellcode von SOS zu erfahren. Dafür müssen sich die SuS mit dem enthaltenen Quellcode beschäftigen und somit ist ein potenziell über die Lerninhalte hinausgehendes Interesse geweckt.

   (c) Die dritte Möglichkeit ist die komplizierteste und ähnelt tatsächlichen Angriffsszenarien. Da das Home-Verzeichnis von Bob von jedem verändert

werden darf, kann Alice dort neue Dateien platzieren. Von besonderem Interesse ist dabei die Datei *.shellrc*, auf die im `intro` Programm (Anhang Seite A-1) in SOS hingewiesen wird. In dieser Datei können Befehle platziert werden, die beim Login des Nutzers Bob ohne dessen Kenntnis ausgeruht werden, um die Datei beispielsweise zu kopieren und somit dem Nutzer *alice* zugänglich zu machen.

Diese Aufgabe soll neben der Anwendung und dadurch Festigung des Erlernten [13], für den Schutz eigener Informationen auf dem eigenen Gerät sowie auf den Computern Dritter sensibilisieren. Außerdem bietet sie die Möglichkeit durch das Entdecken und Ausnutzen realitätsnaher Sicherheitslücken einen Eindruck, zu erhalten wo diese auftreten und wie sie potenziell verhindert werden können.

4 Ausblick

SOS birgt das Potenzial auch den Themenbereich Scheduling im Unterricht veranschaulichend zu begleiten. MentOS unterstützt bereits mehrere vom Lehrplan geforderte Scheduling-Strategien, darunter *Round-Robin*, diverse prioritätsbasierte Verfahren, nur die *First-Come-First-Served* sowie *Shortest-Job-First* Verfahren müssten noch ergänzt werden. Aktuell ist das verwendete Scheduling-Verfahren eine feste Option zur Übersetzungszeit, diese müsste noch durch eine dynamische Option, die zur Laufzeit geändert werden kann, ersetzt werden, um die Unterschiede der Verfahren besser herausarbeiten zu können. MentOS unterstützt bereits mit seiner `SCHEDULER_FEEDBACK` Funktion Informationen über getroffene Scheduling-Entscheidungen über die serielle Schnittstelle zu kommunizieren. Diese Informationen könnten mit einer Visualisierung aufbereitet werden, sodass die SuS anhand der Informationen die Verfahren für unterschiedliche Szenarien untersuchen und anschließend beurteilen könnten.

5 Erklärung des Verfassers

Hiermit versichere ich, dass ich die schriftliche Hausarbeit selbständig verfasst und keine anderen Hilfsmittel als die angegebenen verwendet habe. Die Stellen der Hausarbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind, wurden in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt auch für aus anderen Werken entnommene Zeichnungen, Kartenskizzen und bildliche Darstellungen. Ferner versichere ich, dass ich das Thema weder als Doktor-, Magister-, Diplom-, Master- oder Bachelorarbeit bei einer Hochschule noch als schriftliche Hausarbeit bei einer anderen Staatsprüfung für ein Lehramt behandelt habe.

Nürnberg, den 07. August 2024

Florian Fischer

Akronyme

CWD Current Working Directory. 12

GID Gruppenkennung. 13

MentOS Mentoring Operating System. 9–11, 18, I, A-18

MMU Memory Managment Unit. 3

SOS School Operating System. 1, 4, 6–12, 15–18, I, II, A-18

SuS Schüler und Schülerinnen. 9, 10, 12, 14, 17, 18

UID Benutzerkennung. 13, 14, 16, 17, II, A-18

Literatur

[1] Jens Axboe. *Efficient IO with io_uring*. Version 0.4. 2019. URL: https://kernel.dk/io_uring.pdf.

[2] André Bensoussan, Charles T Clingen und Robert C. Daley. „The Multics virtual memory: Concepts and design“. In: *Communications of the ACM* 15.5 (1972), S. 308–318.

[3] Edward L. Deci und Richard M. Ryan. „The “What” and “Why” of Goal Pursuits: Human Needs and the Self-Determination of Behavior“. In: *Psychological Inquiry* 11.4 (2000), S. 227–268. DOI: 10.1207/S15327965PLI1104_01. eprint: https://doi.org/10.1207/S15327965PLI1104_01. URL: https://doi.org/10.1207/S15327965PLI1104_01.

[4] Peter J Denning. „Virtual memory“. In: *ACM Computing Surveys (CSUR)* 2.3 (1970), S. 153–189.

[5] Hertzsprung at English Wikipedia. *Privilege rings for the x86 available in protected mode*. File: 1024px-Priv_rings.svg.png. 2007. URL: https://en.wikipedia.org/wiki/File:Priv_rings.svg.

[6] ISB. *LehrplanPlus: Fachprofil Informatik*. 2024. URL: https://www.lehrplanplus.bayern.de/fachprofil/gymnasium/informatik.

[7] ISB. *LehrplanPlus: Gymnasium Informatik 11 (NTG)*. 2024. URL: https://www.lehrplanplus.bayern.de/fachlehrplan/gymnasium/11/informatik/ntg.

[8] ISB. *LehrplanPlus: Gymnasium Informatik 12 (erhöhtes Anforderungsniveau)*. 2024. URL: https://www.lehrplanplus.bayern.de/fachlehrplan/gymnasium/12/informatik/erhoeht.

[9] ISB. *LehrplanPlus: Gymnasium Informatik 12 (grundlegendes Anforderungsniveau)*. 2024. URL: https://www.lehrplanplus.bayern.de/fachlehrplan/gymnasium/12/informatik/grundlegend.

[10] Paul Kirschner und Jeroen JG Van Merrienboer. „Ten steps to complex learning: A new approach to instruction and instructional design“. In: *21st century education: A reference handbook*. SAGE Publications Ltd, 2008, S. 244–253.

[11] Josef Leisen. „Praktische Ansätze schulischer Sprachförderung–Der sprachsensible Fachunterricht“. In: *Zugriff am* 29 (2011), S. 2018.

[12] Hilbert Meyer. „Zehn Merkmale guten Unterrichts“. In: *Empirische Befunde und didaktische Ratschläge. Pädagogik* 10 (2003), S. 36–43.

[13] Bruna Fernanda Tolentino Moreira u. a. „Retrieval Practice in Classroom Settings: A Review of Applied Research“. In: *Frontiers in Education* 4 (2019). ISSN: 2504-284X. DOI: 10.3389/feduc.2019.00005. URL: https://www.frontiersin.org/journals/education/articles/10.3389/feduc.2019.00005.

[14] Michael Schroeder und Jerome Saltzer. „A Hardware Architecture for Implementing Protection Rings“. In: *Communications of the ACM* 15.3 (1972), S. 157–170.

[15] Sigrid Schubert u. a. *Didaktik der Informatik*. Springer, 2011.

[16] Livio Soares und Michael Stumm. „FlexSC: Flexible System Call Scheduling with Exception-Less System Calls“. In: *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*. OSDI'10. Vancouver, BC, Canada: USENIX Association, 2010, S. 33–46. URL: https://static.usenix.org/events/osdi10/tech/full_papers/Soares.pdf.

[17] Andrew Tanenbaum. *Modern operating systems*. 4. Aufl. Pearson Education, Inc., 2014.

[18] MentOS Team. *Course Material*. 2024. URL: https://mentos-team.github.io/#course_material.

[19] MentOS Team. *MentOS (Mentoring Operating System)*. 2024. URL: https://github.com/mentos-team/MentOS/blob/24367a58ae4dec9c75f9558ee738781125ef0359/README.md.

[20] LSB Workgroup The Linux Foundation. *Filesystem Hierarchy Standard*. 2015. URL: https://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html.

Schritte des `intro` Programms

Schritt 0

Dieses Programm ist dazu gedacht, Ihnen SOS sowie die Benutzung der Kommandozeile etwas naeherzubringen.
Um SOS zu benutzen muessen, Sie zuerst lernen wie man Programme ausfuehrt. Dazu tippen Sie einfach den Namen des Programms, das Sie ausfuehren moechten, gefolgt von den Argumenten, die dem Programm uebergeben werden sollen, ein und druecken anschliessend die Enter-Taste.
Probieren wir es gleich aus! Tippen Sie *"intro next"* fuer den naechsten Schritt.

Reaktion 0

Sehr gut! Sie haben erfolgreich ein Programm mit Argumenten ausgefuehrt. Alles was die shell, das Programm in dem Sie sich gerade befinden, macht, ist jede Eingabezeile in einzelne Woerter zu zerlegen. Das erste Wort ist das Programm, das ausgefuehrt werden soll. Die restlichen Woerter der Kommandozeile werden an das neue Programm uebergeben. Sie haben das Programm *intro* mit dem Argument *next* aufgerufen. Machen Sie das Gleiche nochmal fuer den naechsten Schritt. Sie koennen das Programm *intro* jeder Zeit wieder aufrufen, um den aktuellen Schritt erneut zu lesen.

Schritt 1

Eine genauere Beschreibung, wie das Betriebssystem Programme ausfuehrt, koennen Sie in *Abschnitt 1.1* des Arbeitsheftes nachlesen. Nachdem Sie nun Progamme ausfuehren koennen, waere es nuetzlich zu wissen, welche Programme es gibt und wie man sie benutzt. Das Programm *man* erlaubt es Ihnen die Dokumentation (engl. manual) des Systems zu lesen. Tippen Sie *"man"* und *<Enter>* um eine Liste aller Dokumentationsseiten zu erhalten. Um eine spezifische Dokumentationsseite zu lesen, fuehren Sie das Programm *man* aus und uebergeben den Namen der Seite als erstes Argument.
Beispiel: man man - zeigt die Dokumentation zu dem Programm man.
Tipp: Viele Programme unterstuetzen auch ein --help Argument.

Frage 1

Aus welcher Datei liest die shell Befehle, bevor sie Nutzereingaben verarbeitet?

Antwort 1

.shellrc

Reaktion 1

Korrekt! Alle Befehle in der Datei .shellrc werden nach dem Einloggen ausgefuehrt.

falls "shell" nicht in Antwort enthalten

Leider nein. *Tipp:* Lesen Sie die Dokumentationsseite zu dem Programm shell.

falls "shell" in Antwort enthalten

Fast. *Tipp:* Lesen Sie die Dokumentationsseite zu shell genau.

Schritt 2

Alle Dateien eines Verzeichnisses (engl. directory) kann man mit dem Programm *ls* (engl. list) anzeigen lassen. Wird kein Pfad zu einem Verzeichnis angegeben, werden die Dateien aus dem *aktuellen Verzeichnis* aufgelistet. Eine Aufgabe des Betriebssystems ist die Verwaltung von Dateisystemen. Anders als in Windows existiert in SOS nur ein Ursprung '/', das
Wurzel-Verzeichnis (engl. *root*). Pfade zu Verzeichnissen oder Dateien koennen ausgehend von diesem Verzeichnis angegeben werden. *Beispiel:*
'/home/alice' ist der Pfad zu Alice Home-Verzeichnis, der Ort aller Dateien von Alice. Pfade, die mit dem Wurzel-Verzeichnis '/' beginnen, nennt man absolute Pfade.

Frage 2

Welche Datei im aktuellen Verzeichnis beginnt mit 'R'?

Antwort 2

README

Reaktion 2

Genau! Der Name ist uebrigens eine Aufforderung.

falls Antwort falsch

Leider falsch. *Tipp:* Benutzen Sie das Programm *ls* um sich alle Dateien auflisten zu lassen.

Schritt 3

Das Betriebssystem merkt sich, in welchem Verzeichnis ein Programm ausgefuehrt wird (engl. *current working directory* kurz *CWD*). Das CWD wird von dem Prozess, der das neue Programm startet, "geerbt". In der shell koennen Sie interaktiv das aktuelle Verzeichnis mit dem Befehl *cd* (engl. change directory) aendern. Die shell zeigt in jeder Zeile das aktuelle Verzeichnis in eckigen Klammern an. Wechslen Sie einige Male das Verzeichnis mit dem *cd* Befehl. *Tipp:* Das Programm *pwd*, (engl. print working directory) zeigt den absoluten Pfad des CWD an.

Frage 3

Welches Symbol zeigt die shell in eckigen Klammern im Verzeichnis /home/alice?

Antwort 3

~

Reaktion 3

Richtig! Das Symbol '~', steht fuer das Home-Vezeichnis des aktuell angemeldeten Benutzers.

falls Antwort falsch

Nein. *Tipp:* Lesen Sie in der Dokumentationsseite zu *cd*, wie Sie ins Home-Verzeichnis gelangen.

Schritt 4

Um an den Inhalt einer Datei zu gelangen, stehen mehrere Programme zur Verfuegung. Das Program *cat* (engl. conccatinate) beispielsweise gibt den Inhalt einer oder mehrerer Dateien zusammenhaengend aus. *head* kann verwendet werden, um nur die ersten Zeilen von Dateien anzeigen zu lassen. Ist eine Datei zu lang, um auf den Bildschirm angezeigt zu werden, kann *more* verwendet werden, um die Datei Zeile fuer Zeile zu lesen. Um gezielt nach Woertern zu suchen, steht das Programm *fgrep* zur Verfuegung.

Frage 4

Wie lautet das erste Wort in der dritten Zeile der Datei */etc/passwd*?

Antwort 4

bob

Reaktion 4

Korrekt! In der Datei /etc/passwd werden alle Benutzerzugaenge des Systems aufgelistet. Mehr Informationen koennen Sie in der Dokumentationsseite zu *passwd* nachlesen.

falls Antwort falsch

Das ist so nicht richtig

Schritt 5

Wird eine Datei mehrmals benoetigt, kann sie mit dem Programm *cp* (engl. copy) an einen neuen Ort bzw. in eine Datei mit anderem Namen kopiert werden. In der Angabe von Pfaden koennen die besonderen Bezeichner *"."* und *".."* verwendet werden. Besonders um relative Pfade, also Pfade ausgehend vom aktuellen Verzeichnis (CWD) anzugeben, koennen diese nuetzlich sein. *"."* steht dabei fuer das Verzeichnis selbst und *".."* bezeichnet das Oberverzeichnis.

Beispiel: CWD=/home/alice

. ⇒ /home/alice

.. ⇒ /home

../bob ⇒ /home/bob

Frage 5

Wie lautet der Befehl, um die Datei namens "foo" in die Datei "bar" im Oberverzeichnis zu kopieren?

Antwort 5

cp foo ../bar

Reaktion 5

Korrekt! Sehr schoen.

falls Antwort nicht ".." enthält

Leider falsch. Den Bezeichner fuer das Oberverzeichnis nicht vergessen.

falls Antwort falsch

Leider falsch.

Schritt 6

Wurde eine Datei versehentlich kopiert oder wird nicht mehr benoetigt, kann sie mit Hilfe des Programms *rm* (engl. remove) entfernt werden. Leere Verzeichnisse lassen sich mit dem Programm *rmdir* (engl. remove directory) entfernen.

Frage 6

Wie lautet der Befehl, um die Datei "todo" des Nutzers bob aus dessen Home-Verzeichnis, unabhaengig vom aktuellen Verzeichnis zu entfernen?

Antwort 6

rm /home/bob/todo

Reaktion 6

Stimmt genau!

falls Pfad der Antwort nicht mit "/" startet

Achten Sie darauf einen absoluten Pfad anzugeben.

falls Antwort nicht "home" enthält

Home-Verzeichnisse befinden sich unter /home/

falls Antwort nicht "bob" enthält

Die Datei soll aus bobs Home-Verzeichnis geloescht werden.

Schritt 7

Die shell erlaubt es mit dem ">"-Operator die Ausgabe eines Programms in eine Datei umzuleiten. Jedem Programm stellt das Betriebssystem zwei Ausgabe-Kanaele zur Verfuegung, die normalerweise einfach auf dem Bildschirm erscheinen. Sie werden *stdout* (standart output) und *stderr* standart error) genannt. *Beispiele*: Um stdout des Progamms ls in die Datei "datei-liste.txt" umzuleiten, kann der Befehl "*ls > datei-liste.text*" verwendet werden. Um nur die Fehler des Programms rm in die Datei "remove-errors.txt" umzuleiten, kann der Befehl *"rm foo bar 2> remove-error.txt"* verwendet werden. Um beide Kanaele eines Programms umzuleiten, kann der Befehl "*programm &> ausgaben.txt*" verwendet werden. Erstelle die Datei "/home/alice/hello.txt", die nur das Wort "hello" enthaelt mit Hilfe des *echo* Programms.

Lösung 7

echo hello > hello.txt

Reaktion 7

Perfekt. Sie sind bereit!

falls Datei hello.txt nicht lesbar ist

Die Datei /home/alice/hello.txt existiert noch nicht.

falls die Datei weniger als fünf Zeichen enthält

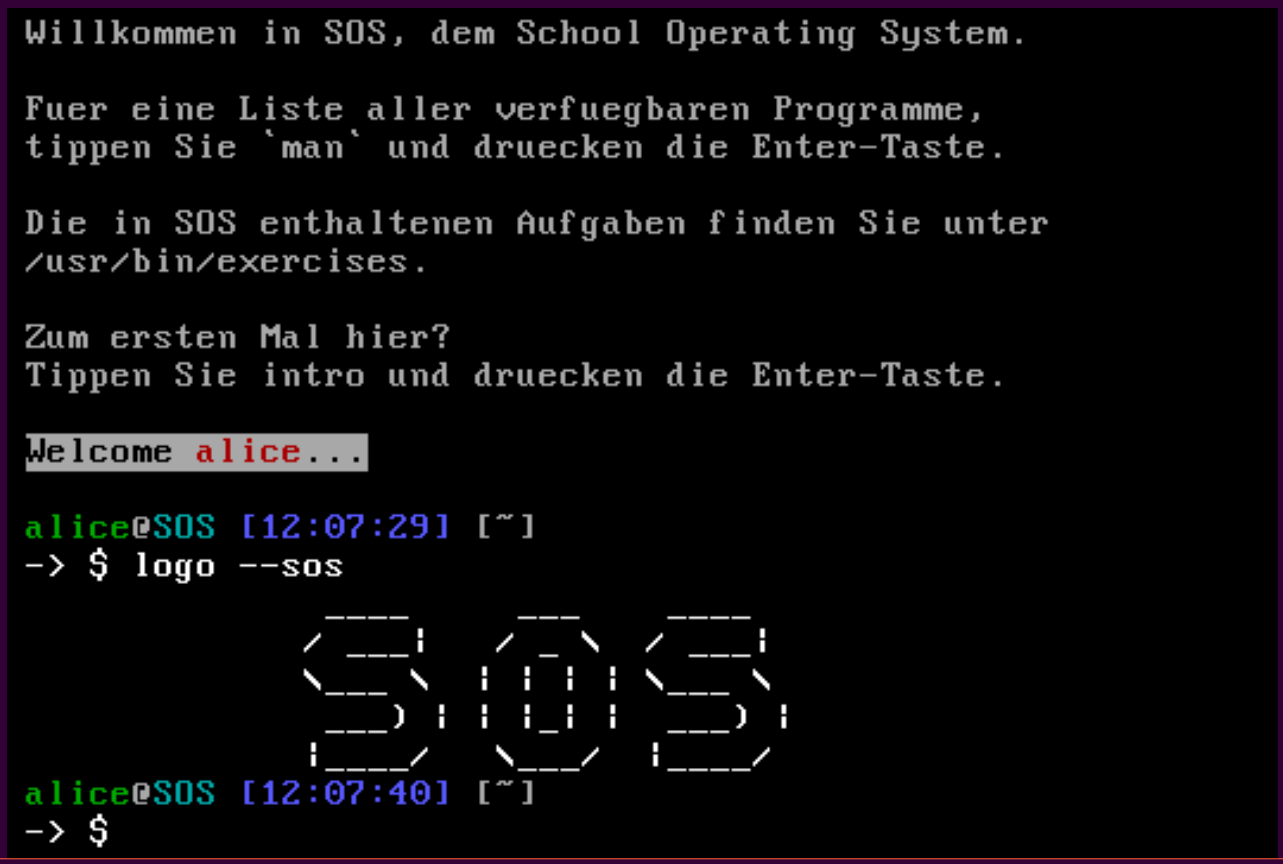
Die Datei enthaelt zu wenig Text.

falls die Datei nicht mit hello beginnt

Die Datei enthaelt nicht den Text "hello".

SOS - Hilfe, was ist hier los?

Anleitung zum School Operating System.



Arbeitsheft zum Thema Betriebssysteme

Inhaltsverzeichnis

Konzeption des Arbeitshefts:

Das Arbeitsheft begleitet den Einsatz des School Operating System (SOS) im Unterricht, das die Grundlagen und Aufgaben von Betriebssystemen nach dem LehrplanPlus der 12. Jahrgangsstufe mit erhöhtem Anforderungsbereich behandelt.

Florian Fischer
Hans-Sachs-Gymnasium Nürnberg
E-Mail: florian.fischer@muhq.space

Dieses Arbeitsheft ist lizenziert unter einer Creative Commons CC BY-SA Lizenz (Namensnennung – Weitergabe unter gleichen Bedingungen).

Über SOS und dieses Arbeitsheft

Dieses Arbeitsheft kann als Begleitung oder als Inspiration in Kombination mit SOS im Unterricht eingesetzt werden.

School Operating System (SOS)

Bei SOS handelt es sich um eine Kombination aus einer portablenn virtuellen Maschine (QEMU) und dem von der Universität Verona entwickelten und für SOS und den Informatikunterricht angepassten Betriebssystem MentOS.

Ziele

Betriebssysteme sind in unserem digitalen Alltag allgegenwärtig. Dennoch ist dieser Fakt den meisten Anwenderinnen nicht bewusst, was dafür spricht, dass moderne Betriebssysteme ihrer Aufgabe nachkommen, die einfache und sichere Benutzung der unterschiedlichsten Geräte von Smartwatches, über Mobiltelefone zu klassischen PCs reibungslos zu ermöglichen.

SOS dient dazu, die oft schwer erfahrbaren Aufgaben von Betriebssystemen erlebbar zu machen. Es werden die wichtigsten und universellen Begriffe im Kontext von Betriebssystemen wiederholt und veranschaulicht. Besonderer Fokus liegt dabei auf der Gewährleistung eines sicheren Betriebs durch die Verwaltung und Durchsetzung von Berechtigungen im Dateisystem.

Außerdem lädt MentOS durch seinen verhältnismäßig einfachen Aufbau, sowie die vorhandene Dokumentation dazu ein, Neues zu entdecken, weiterführendes Interesse zu wecken und in gewissen Umfang auch zu befriedigen.

Benutzung

SOS ist so konzipiert, dass die Benutzung im Unterricht möglichst einfach ist.

1. Eine einsatzfähige Version von SOS inklusive aller Arbeitsmaterialien kann unter https://github.com/fischerling/SOS/releases direkt heruntergeladen werden. Die Datei `sos-win.zip` enthält alle benötigten Komponenten zum Einsatz von SOS.
2. Zur Benutzung von SOS muss das heruntergeladene Archiv komplett extrahiert werden.
3. Um das Betriebssystem in der enthaltenen virtuellen Maschine zu starten, reicht es die Datei `run.bat` durch Doppelklick auszuführen. Gegebenenfalls muss eine Sicherheitswarnung von Windows, bestätigt werden.
4. Nachdem MentOS in der virtuellen Maschine gestartet ist, wird die Nutzenden nach einem Login, wie in Abbildung 1 dargestellt, gefragt. Alle Aufgaben in SOS sind für den Benutzer `alice` mit Passwort `alice` ausgelegt.

Tipp: Um SOS leichter Benutzen zu können, bietet es sich an die Schriftgröße in QEMU unter *View* zu vergrößern. Außerdem kann der potentiell gefangene Mauszeiger in QEMU durch Drücken von `Strg + Alt + G` wieder befreit werden.

```
Creating init process...                                        [OK]
Initialize floating point unit...                               [OK]
Initialize signals...                                           [OK]
Username: alice
Password:
```

Abbildung 1: Login-Aufforderung von MentOS

1 Grundlagen und erste Schritte

Betriebssystem

Das Betriebssystem ist im vereinfachtem Sinn die Software, die im privilegierten *Kernelmodus* der Hardware ausgeführt wird, siehe Abbildung 2. Dies muss aber nicht immer der Fall sein. Grundsätzlich hat das Betriebssystem die Aufgabe die bestehende Hardware den Nutzerprogrammen in geeigneter Weise zur Verfügung zu stellen und deren sichere Verwendung zu gewährleisten.

Der *Kern* des Betriebssystems, das zentrale Programm, das alle notwendigen privilegierten Operationen ausführt, wird englisch *kernel* genannt. Man unterscheidet unterschiedliche Arten von *Betriebssystemkernen* anhand der in ihnen enthaltenen Funktionen und wie ihre Dienste den anderen Schichten angeboten werden. *Monolithische kernel*, in denen alle Funktionen des Betriebssystems in einem Programm im privilegierten Modus der Hardware umgesetzt werden, haben sich im Alltag aus Effizienzgründen durchgesetzt.

Die Hauptaufgabe des Betriebssystems ist demnach die Verwaltung und Bereitstellung der vorhandenen Hardwareressourcen, wie der Rechenzeit auf den Rechenkernen, den Hauptspeicher, sowie Zugriff auf Peripherie-Geräte wie das Netzwerk, Festplatten oder Ein-/ und Ausgabegeräte.

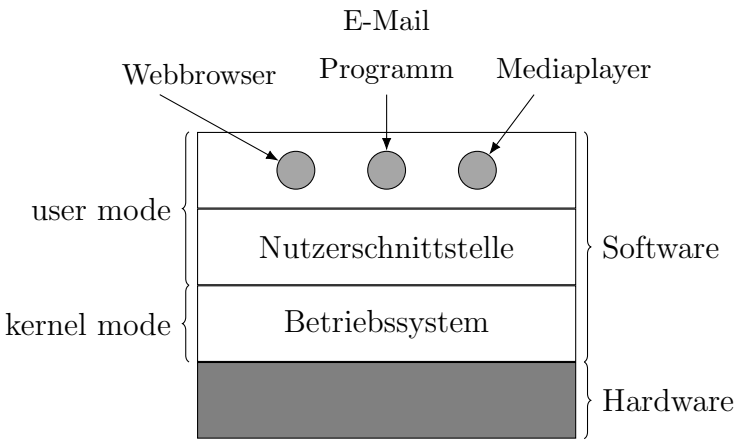


Abbildung 2: Platz des Betriebssystems im Stapel der Technologien.

Dabei muss das Betriebssystem, die Sicherheit der Ressourcen (englisch *security*) gegen bösartige Dritte, andere Nutzer auf dem System, Schadsoftware, Angriffe über die angeschlossenen Peripherie-Geräte, zumeist das Netzwerk, sicherstellen.

Darüber hinaus ist es auch Aufgabe des Betriebssystems, die Betriebssicherheit (englisch *safety*) zu gewährleisten, dass heißt, beispielsweise den Verlust von Daten oder das Abstürzen des Systems bei einem Fehler zu verhindern. Auch muss der Zugriff auf die Hardware zentral koordiniert werden, um beispielsweise die fehlerhafte gleichzeitige Nutzung der Festplatte durch mehrere Nutzerprogramme zu verhindern.

Um diese Schutzziele zu erfüllen, arbeitet das Betriebssystem eng mit der verwendeten

Hardware zusammen. Diese erlaubt es beispielsweise durch die Trennung der Ausführung in *Benutzermodus* und *Kernelmodus*, gewisse Operationen auf das Betriebssystem einzuschränken. Darüber hinaus sorgt die Hardware nach Instruktion durch das Betriebssystem dafür, dass Programme nur auf ihren eigenen exklusiven Speicherbereich und nicht den anderer Programme zugreifen können. Das schützt sowohl das gesamte System vor Fehlern in einzelnen Programmen, sowie das einzelne Programm vor dem böswilligen Zugriff Dritter.

Arbeitsauftrag: Los gehts!

Um den Umgang mit dem rein textbasierten SOS zu üben und einige grundlegende Konzepte aus der Welt der Betriebssysteme zu erkunden, stellt SOS ein `intro` Programm zur Verfügung.

1. Starten Sie, falls noch nicht geschehen, SOS durch Ausführen der Datei `run.bat`.
2. Melden Sie sich als der Benutzer `alice` mit dem Passwort `Alice` an.
3. Starten Sie nach einer erfolgreichen Anmeldung, wie in Abbildung 3 gezeigt, das `intro` Programm durch Eintippen des Namens und Drücken der Enter-Taste.

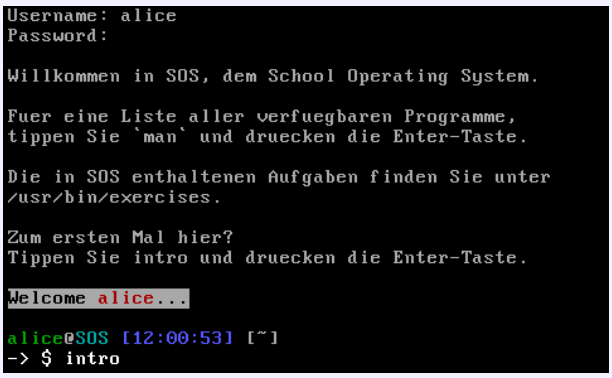


Abbildung 3: Starten des `intro` Programms, nach erfolgreicher Anmeldung.

Im Folgenden werden die Konzepte in der Reihenfolge, in der sie im Ablauf des `intro`-Programms auftauchen, beschrieben.

1.1 Prozesse

Eines der fundamentalsten Konzepte eines Betriebssystems ist der *Prozess*. Ein Prozess repräsentiert ein laufendes Programm. Diese Abstraktion erlaubt es, mehrere Programme scheinbar gleichzeitig auf nur einem Rechenkern auszuführen.

Prozess

Ein Prozess besteht aus allen Datenstrukturen, die benötigt werden, um ein Programm auszuführen, beispielsweise einem exklusiven Speicherbereich, sowie dem auszuführenden Programmcode. Jeder Prozess hat einen *Zustand*, anhand dessen er in der Ablaufplanung des Betriebssystems, berücksichtigt wird. Wartet ein Prozess auf eine Ressource, wie beispielsweise Daten von der Festplatte, wird er von der Ablaufplanung ausgenommen bis die Ressource verfügbar ist.

Um die Sicherheit des Systems zu gewährleisten, hat jeder Prozess zugewiesene Berechtigungen, herkömmlich und so auch in SOS sind die Berechtigungen mit dem *Besitzer* und der Gruppe des Prozesses verbunden. Der *root* Nutzer, mit der Benutzerkennung (UID) 0 hat in UNIX-ähnlichen Systemen uneingeschränkte Rechte, vergleichbar mit dem *Admin* Nutzer eines Windows System. Die UID und Gruppenkennung (GID) eines Prozesses werden beim Erstellen des Prozesses vom *Elternprozess* geerbt. Da ein Prozess immer nur von einem anderen Prozess erzeugt werden kann, bilden alle Prozesse des Systems einen Baum. Wobei die Wurzel des Baums, der `init`-Prozess mit Prozessnummer (PID) 1 vom Betriebssystem gestartet wird.

PROZESS
Zustand
Besitzer
Argumente
CWD
Elternprozess
Ausgabe-Streams

Abbildung 4: Vereinfachte Klassenkarte eines Prozesses

Die *Kommandozeile* ermöglicht die rein textbasierte Benutzung des Computer, das heißt die Erzeugung und Kontrolle von Prozessen. Ein `shell` genanntes Programm, liest Zeile für Zeile Befehle des Nutzers ein. Jede Zeile wird dabei anhand der enthaltenen Leerzeichen getrennt.

Programme werden anhand ihres Namens in Verzeichnissen, im `Suchpfad` (UNIX: `$PATH`; Windows: `Path`) gesucht.

Das erste Wort der Zeile wird als Name eines Programms interpretiert. Es wird ein neuer Prozess gestartet, dieser führt das Programm, das anhand des Namens gefunden wurde, aus. Der Rest der Zeile wird dem neuen Programm als *Argumente* übergeben.

Jeder Prozess hat ein Verzeichnis zugewiesen, in dem er aktuell „arbeitet“ und von dem aus alle relativen Dateipfade interpretiert werden. Dieses „Arbeitsverzeichnis“ wird englisch Current Working Directory (CWD) genannt. Das CWD wird von dem Prozess, der den neuen Prozess startet, dem *Elternprozess* geerbt. In der shell kann das CWD mit dem Befehl `cd` (englisch change directory) geändert werden.

1.2 Dateien

Ein weiteres essentielles Konzept nahezu aller Betriebssysteme ist die *Datei*. Um die Eigenheiten verschiedener persistenter Speichermedien, von Festplatten, über USB-Sticks bis hin zu Magnetbändern, zu verstecken, bieten Betriebssysteme das Konzept von Dateien an. Um Dateien zu strukturieren werden besondere Dateien sogenannte Verzeichnisse (englisch *directory*) verwendet. Ähnlich zu Prozessen können Dateien in einem Dateisystem in einer Baumstruktur gruppiert werden (vergleiche Abbildung 6).

Datei

Die gängigste Umsetzung von Dateien ist sie als Folge von Bytes bekannter *Größe* zu behandeln. Der Inhalt einer Datei hat für das Betriebssystem normalerweise keine Bedeutung und muss von Nutzerprogrammen selbst interpretiert werden. Dateien unterscheiden sich zum Speicherbereich eines Prozesses dadurch, dass sie nach der Beendigung des Prozesses weiter existieren und von mehreren Prozessen und sogar nach einem Neustart des Systems verwendet werden können. Um Dateien unterscheiden und wiederverwenden zu können, ist jeder Datei ein *Name* zu geordnet. Manche Betriebssysteme schränken die erlaubten Zeichen in Dateinamen ein.

Abbildung 5: Vereinfachte Klassenkarte einer Datei

Wie bereits erwähnt, existieren in einem Betriebssystem unterschiedliche Arten von Dateien. Neben regulären Dateien, die eine Folge von Bytes speichern und Verzeichnissen, die die Namen anderer Dateien, die in diesem Verzeichnis enthalten sind, speichern, werden auch Hardware Geräte in UNIX-ähnlichen Systemen als Dateien bereitgestellt. Das hat den Vorteil, dass die bestehenden Mechanismen der Dateiverwaltung, wie die Zugriffskontrolle, auch auf Hardware Geräte, wie USB-Sticks, angewendet werden können.

Da in einem System mehrere Dateien mit dem gleichen Namen existieren können, wird um eine bestimmte Datei verwenden zu können, nicht nur ihr Name sondern auch ihr Ort im Dateisystem benötigt.

absolut:
`/home/alice/README`
relativ:
`README`
`../bob`

Der Ort einer Datei kann auf zwei Arten angeben werden als **absoluter** Pfad oder **relativer** Pfad. Ein Pfad besteht aus einer beliebigen Liste an Verzeichnissen und einem optionalen abschließenden Dateinamen, getrennt von *Pfadseparatoren*. Der Pfadseparator in SOS ist wie in allen UNIX-ähnlichen Systemen (beispielsweise Linux oder MacOS) das Zeichen `/`. Windows benutzt das Zeichen `\` als Pfadseparator.

Absolute Pfade beginnen immer mit dem Wurzelverzeichnis des Dateisystems (`/` in SOS).

Relative Pfade werden ausgehend vom aktuellen Arbeitsverzeichnis CWD des Prozesses interpretiert.

Jedes Verzeichnis enthält die beiden besonderen Einträge `.` und `..`, die zusätzlich zu Namen in Pfaden vorkommen können. Der besondere Eintrag `.` steht dabei für das Verzeichnis selbst und `..` für das Oberverzeichnis.

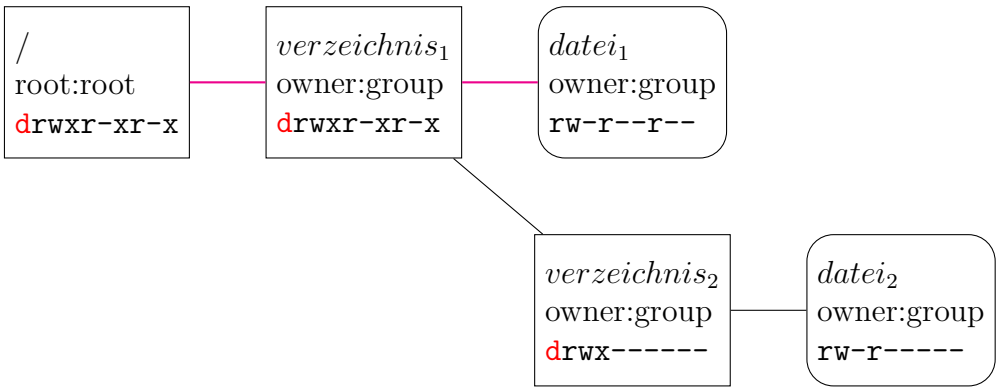


Abbildung 6: Darstellung der Baumstruktur eines Dateisystems ausgehend vom Wurzelverzeichnis `/`. Ob es sich bei einer Datei um ein Verzeichnis handelt, wird in der Darstellung der Berechtigungen mit einem vorangestellten d symbolisiert. Der absolute Pfad zu $datei_1$ lautet `/verzeichnis1/datei1`.

2 Datei-Berechtigungen

Um die beiden Sicherheits-Schutzziele *security* und *safety* zu gewährleisten, haben Dateien einen *Besitzer* und eine feste Zuweisung an *Berechtigungen*. Diese Berechtigungen werden bei der Verwendung einer Datei überprüft und die Operation gegebenenfalls vom Betriebssystem abgelehnt.

Datei-Berechtigungen

Eine sehr rudimentäre, aber dennoch in UNIX-ähnlichen System weit verbreitete Art, die Berechtigungen einer Datei zu verwalten, ist diese in neun Bit zu speichern. Diese Bits werden, wie Abbildung 7 dargestellt, in drei Klassen á drei Bits aufgeteilt. Die Klassen stehen für den Besitzer der Datei, die Gruppenkennung der Datei, sowie alle anderen.
Das erste Bit (read) innerhalb einer Klasse gibt an, ob die Datei gelesen werden darf. Das zweite Bit (write) regelt, ob die Datei geschrieben werden darf. Das dritte Bit (exute) steuert, ob die Datei in einem Prozess ausgeführt werden darf, oder falls es sich um ein Verzeichnis handelt, ob auf andere Dateien innerhalb des Verzeichnisses zugegriffen werden darf.
Vor der Verwendung einer Datei wird zunächst die Klasse des Prozesses, der die Datei verwenden möchte, bestimmt und anhand derer geprüft, ob die Operation erlaubt ist.

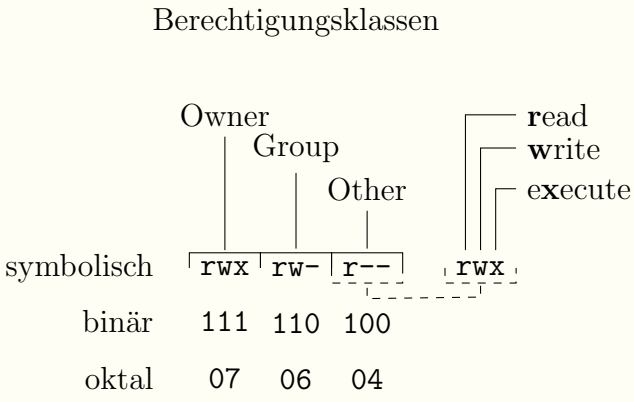


Abbildung 7: Berechtigungs-Bits einer Datei gegliedert nach den Berechtigungsklassen

Neben den neun grundlegenden Bits für die drei Klassen existieren noch drei zusätzliche

Bits für erweiterte Dateiberechtigungen. Bei dem ersten Bit handelt es sich um das sogenannte „*sticky bit*“, das bei von allen schreibbaren Verzeichnissen, das Löschen verhindert. Die zwei verbleibenden Bits werden „*SUID*“ (Set UID) beziehungsweise „*SGID*“ (Set GID) genannt. Ist eines dieser Bits gesetzt, wird das Programm mit dem Besitzer beziehungsweise der Gruppe der Programm-Datei statt des Elternprozesses ausgeführt. Dies wird verwendet, um beispielsweise Nutzern kontrollierten Zugriff auf den *root*-Nutzer mit UID 0 zu gewähren. Das in SOS enthaltene *SUID*-Programm `doas` prüft anhand einer Konfigurationsdatei, ob der aktuelle Benutzer die Berechtigung hat, Befehle als *root*-Nutzer auszuführen, und führt gegebenenfalls den als Argumente übergebenen Befehl mit UID 0 aus.

Moderne Dateisysteme unterstützen darüber hinaus, eine spezifischere Steuerung der Berechtigungen auf Basis einer Zugriffssteuerungsliste (ZSL).

Arbeitsauftrag: Berechtigungsbits

Gängige Berechtigungen für Verzeichnisse sind die Bits `0755`. Wandeln Sie die Bits aus der Oktalschreibweise in ihre symbolische Darstellung um und interpretieren Sie ihre Bedeutung.

Untersuchen Sie mit dem Programm `stat` die Berechtigungen der folgenden Dateien:

1. `/bin/doas`
2. `/bin/ls`
3. `/etc/passwd`
4. `/etc/shadow`
5. `/home/bob`

2.1 Kartographie der Beispiellandschaft

SOS enthält eine Verzeichnisstruktur, in der die Bedeutung der Berechtigungen von Dateien und Verzeichnissen an Beispielen aus der analogen Welt verdeutlicht werden. Diese wird beim Start der `file-permission` Aufgaben durch Ausführen des Programms `/usr/bin/exercises/file-permissions/setup` angelegt.

Arbeitsauftrag: Kartographie der Beispiellandschaft

Erkunden Sie die Verzeichnisstruktur unter `/home/alice/Landschaft`. Zeichnen Sie die Beispiellandschaft als Baumdiagramm (vergleiche Abbildung 6). Stellen Sie Vermutungen darüber an, warum zu einem Beispiel die bestehenden Berechtigungen passen. Hilfreiche Programme: `ls`, `stat`, `cd`.

2.2 Kartographie des Dateisystems

Die Verzeichnisstruktur von SOS, die sich am File Hierarchy Standard für UNIX-ähnliche Systeme orientiert, enthält ganz bestimmte Berechtigungen für gewisse Teilbäume und Dateien. Eine kurze Erklärung zu ausgewählten Verzeichnissen kann in der in SOS enthaltenen Dokumentationsseite `hier` nachgelesen werden.

Arbeitsauftrag: Kartographie des Dateisystems

Erstellen Sie ein Baumdiagramm des Dateisystems von SOS. Färben Sie die Teilbäume entsprechend ihres Besitzers ein und achten Sie dabei auch auf Ausreißer, die von den anderen Berechtigungen abweichen und stellen Sie Vermutungen an, warum dies jeweils der Fall ist.

2.3 Herausforderung: Geheimnisse

Oft sind nicht akkurat gepflegte Berechtigungen Ursache für Sicherheitsprobleme. Diese erlauben es zum Beispiel Unbefugten sensible Informationen zu lesen oder in schlimmsten Fall beliebigen Code auszuführen. SOS enthält eine kleine Herausforderung, bei der die Geheimnisse des Nutzers `alice` geschützt und die Geheimnisse des Nutzers `bob` herausbekommen werden müssen.

Arbeitsauftrag: Geheimnisse

Nach dem Start der `file-permission` Aufgabe durch Ausführen des Programms `/usr/bin/exercises/file-permissions/setup` enthält das Verzeichnis `/home/alice` die Datei `secrets.txt`, die es zu schützen gilt. Das Verzeichnis `/home/bob` enthält die Dateien `secrets.txt` und `top_secret.txt`, deren Inhalt nötig ist, um die Fragen des Programms `/usr/bin/exercises/file-permissions/checkup` erfolgreich zu beantworten. Reflektieren Sie die Art und Weise, wie Sie an die Informationen gelangt sind und ob dies verhindert werden kann.

Happy Hacking :)

Dem Autor sind drei Möglichkeiten bekannt, den Inhalt der Datei `top_secret.txt` zu erfahren. Der ersten Person die alle drei, oder potenziell noch weitere Möglichkeiten findet und diese unter sos@muhq.space erklärt, verspricht der Autor hiermit ein Heißgetränk Ihrer Wahl.

Akronyme

CWD Current Working Directory. 4, 5

GID Gruppenkennung. 4

PID Prozessnummer. 4

SOS School Operating System. i, 1, 3–5, 7–10

UID Benutzerkennung. 4, 7

ZSL Zugriffssteuerungsliste. 7

In MentOS aufgenommene Beiträge, die im Rahmen von SOS entstanden sind:

- Trennung von realer und effektiver UID eines Prozesses und Implementierung der benötigten Systemaufrufe.
- Implementierung der Systemaufrufe und Nutzerprogramme um die Attribute von Dateien zu verwalten
- Ergänzung von Nutzerprogrammen die zur Bearbeitung der im Arbeitsheft enthaltenen Aufgaben benötigt werden
- Ausführen von Skripten mit Hilfe eines Interpreters
- Vervollständigung der enthaltenen Dokumentationsseiten
- Implementierung von Ausgabeumleitung und den dazu benötigten Systemaufrufen
- Unterstützung der *SUID* und *GUID* bits
- Ergänzung des deutschen Tastaturlayouts
- Korrekte Auflösung von Datei Pfaden
- Korrekte Überprüfung von Dateiberechtigungen
- Behebung diverser Fehler
- Implementation einer Testinfrastruktur

Ergänzungen die nur in SOS enthalten sind:

- Setzen realistischer Dateiberechtigungen im kompletten Dateisystem
- Einführung der beiden Nutzer *alice* und *bob*
- Bereitstellung der Quelldateien von SOS im Dateisystem
- Implementierung der Nutzerprogramme `doas`, `cp`, `head`, `fgrep`, `apropos`
- Umsetzung des interaktiven `intro` Programms, sowie der Dateiberechtigungsaufgabe.