Fabian Hupfeld (35523329) und Konstantin Stitz (35532242)

Übungsaufgaben 1

Aufgabe 1.)

a.)

- Das Betriebssystem wird für die Programmausführung benötigt, denn es sorgt dafür, dass Programme geladen, gestartet und beendet werden können

- Ressourcen des Computers wie der Speicher, Prozessor, Ein- und Ausgabegeräte etc. werden vom Betriebssystem verwaltet und den Programmen zugeteilt, dies ist nötig, damit Konflikte vermieden werden und die Ressourcen effizient und sinnvoll ausgelastet/ benutzt werden

- Das Betriebssystem verhindert mit seinem Zugriffsschutz, dass nicht erlaubte oder nicht gewollte Zugriffe (z.B. eines Programmes auf einen bestimmten Speicherbereich) verhindert werden

- Das Betriebssystem wird benötigt, damit der Rechner mit anderen Rechnern, zum Beispiel via E-Mail, kommunizieren kann, es stellt also Schnittstellen für Programme bereit, damit dies möglich ist.

b.)

Multiprogramming bedeutet, dass das Betriebssystem gleichzeitig mehrere Arbeitsaufträge (Jobs) im Speicher haben kann und die CPU diese abwechselnd bearbeiten kann(also schnell zwischen ihnen umschalten kann), wodurch die Effizienz erhöht werden kann, da die CPU immer etwas abzuarbeiten hat, während sie ohne Multiprogramming unter Umständen darauf warten muss bis der nächste Job in den Speicher geladen ist. Time-Sharing dagegen bedeutet, dass mehrere Benutzer gleichzeitig auf einem Computer arbeiten können. Man hat also einen Computer, auf den mehrere Benutzer von verschiedenen Zugangspunkten zugreifen können. Dies hat den Vorteil, dass die CPU effizienter ausgelastet wird, da in Leerlaufzeiten eines Benutzers Befehle/Berechnungen von anderen Benutzern (mithilfe von Multiprogramming) ausgeführt werden können.

Aufgabe 2.)

a.) falsch, Mikroprogamme sind sehr hardwarenahe Programme, die dazu dienen, auf unterschiedlichen Rechnersystemen eine einheitliche Ausführungsumgebung zu haben

b.) falsch, Multiprogramming bedeutet lediglich das im Speicher mehrere Jobs gleichzeitig vorgehalten werden, damit der Prozessor schnell zwischen diesen wechseln kann und während der Ein-/Ausgaben und/oder Schreib-/Lesevorgängen keinen Leerlauf hat. Dazu reicht jedoch auch ein Prozessor

c.) wahr, die Betriebssysteme ermöglichen dann, dass eine Kommunikation der Rechner trotzdem möglich ist (über die Middleware-Komponenten)

d.) wahr, da das Betriebssystem für die Verwaltung der Rechnerressourcen wie z.B. Arbeitsspeicher verantwortlich ist, würde es bei 2 Betriebssystemen gleichzeitig zu Konflikten kommen, wer wann was benutzen darf. Die Einzige Möglichkeit wäre eine Virtualisierungssoftware, die einen 2. oder mehrere weitere Rechner simuliert, man benötigt dann jedoch immer noch genau ein Hostbetriebssystem, auf welchem die Virtualisierungssoftware läuft

e.) wahr, da diese Ressourcen sind, auf die das Betriebssystem zugreifen und diese Verwalten/ den Programm zuteilen kann

Aufgabe 3.)

b.)

mkdir, um ein neues Verzeichnis zu erstellen

open, um die Datei zu öffnen, oder sie zu erstellen falls sie nicht vorhanden ist

write (<<), um in die geöffnete Datei zu schreiben

close, um den file descriptor der Datei zu schließen

c.)

Der Grund für die Unterteilung in Kernel und User Mode ist, dass nicht jede Anwendung vollen Zugriff auf das gesamte System haben soll, durch die Unterteilung wird also die Sicherheit erhöht. Im Kernel laufen dann nur die für das Betriebssystem wichtigsten und vertrauenswürdigsten Programme, während normale Programme im User Mode laufen. Diese können dann, zum Beispiel, nicht einfach Speicher verändern, sondern müssen eine Anfrage an das Betriebssystem stellen und der Kernel muss das dann ausführen. Die Unterteilung ist also ein wichtiges Sicherheitskonzept und Zugriffsschutz. Zudem stürzt nicht der gesamte Computer ab, wenn ein Programm im User Mode abstürzt, würde es mit Kernel Rechten abstürzen wäre das für den Computer wesentlich kritischer.

d.)

Der call – Befehl sorgt dafür, dass die Adresse die nach dem call-Befehl folgt im Stack gespeichert wird und springt danach in das Unterprogramm welches durch die Zieladresse angegeben wird. Dieses kann mit ret beendet werden und dann mit der im Stack gespeicherten Adresse fortgeführt werden.

Der mov-Befehl (Move Data) sorgt dafür das der erste übergebene Operand in den zweiten Operanden kopiert wird.

Der leave-Befehl macht alle Änderungen am Stack nach Ablauf eines Unterprogrammes rückgängig, bevor zurückgesprungen wird. Er entspricht der Nacheinander Ausführung von mov esp, ebp und pop ebp.

Der ret-Befehl (Return from Near-Subroutine) sorgt dafür, dass der Inhalt des IP-Registers, was vorher auf dem Stack, zB. Durch call, abgespeichert wurde wiederhergestellt wird. Somit wird das durch call aufgerufen Unterprogramm beendet und mit dem normalen Ablauf fortgefahren.