Vorlesung Betriebssystemarchitektur SS 2006
Aufgabenblatt 5 vom 6. Juli 2006
(Vorstellung der Lösungen bei den Tutoren bis zum 20. Juli 2006)

Aufgabe 5.1: (10 Punkte)
Gegeben sei ein 32-Bit-Rechnersystem. Die Seitenrahmengröße (Page-Frame Size) beträgt 1Kbyte. Das Betriebssystem unterstützt maximal 4Gbyte physikalischen Speicher. Die virtuelle Speicherverwaltung unterstützt lediglich eine Ebene von Page-Tables, d.h. zu jedem Prozess gehört genau eine Tabelle, die den gesamten virtuellen Speicher auf den physikalischen abbildet. Diese Tabellen können nicht ausgelagert werden. Der Rechner hat insgesamt 64Mbyte physikalischen Speicher. Das Betriebssystem benötigt mindestens 3Mbyte Speicher, der nicht ausgelagert werden kann.
Beantworten Sie Ihrem Tutor folgende Fragen:

- Wie viele Einträge enthält eine Page-Table?
- Wie viel physikalischer Speicher wird von einer Page-Table belegt? Jeder Page-Table Eintrag soll die Page-Frame Nummer und 2 Kontroll-Bits enthalten.
- Wenn zwei Prozesse in diesem System laufen, wie groß kann das Working-Set eines Prozesses maximal sein?

Aufgabe 5.2: (15 Punkte)
Erläutern Sie Ihrem Tutor die Rollen der *Standby-*, *Modified-*, *Free-* und *Zero page-* Listen in der Windows Speicherverwaltung! Welche Eigenschaften haben Seiten, die in den einzelnen Listen enthalten sind?

Aufgabe 5.3: (15 Punkte)
Beantworten Sie Ihrem Tutor folgende Fragen:

- Wie ist das Konzept der Zugriffs Kontrolle in Windows implementiert?
- Was ist ein *Security Descriptor*?
- Was ist der typische Aufbau einer DACL?
- Was ist der Zweck der SACL?

Aufgabe 5.4: (10 Punkte)

- Ein Programm öffnet erfolgreich eine Datei mit Hilfe von CreateFile() und der „GENERIC_WRITE“ Option. Im nächsten Schritt modifiziert ein Benutzer die Zugriffsrechte für diese Datei: Er ergänzt „deny ACE“ und sperrt damit den Schreibzugriff für alle Nutzer. Danach versucht das oben genannte Programm mit Hilfe von WriteFile() Daten in die Datei zu schreiben.
  Wird der WriteFile()-Aufruf ohne Fehlermeldung ablaufen? Begründen Sie Ihre Antwort!
- Alice ist ein Mitglied der Gruppe „Student“ und möchte auf eine Datei zugreifen der folgende DACL mit den aufgeführten ACEs zugeordnet ist:
  - 1. Bob darf nicht lesend auf die Datei zugreifen.
  - 2. Studenten dürfen nicht lesend auf die Datei zugreifen.
  - 3. Alice darf schreibend zugreifen und die Datei löschen.

  Kann Alice lesend auf die Datei zugreifen? Was würde sich ändern, wenn die zweite ACE entfernt wird? Begründen Sie Ihre Antwort!
- Angenommen es existiert ein Programm „shutdown.exe“ das den Rechner herunterfährt. Wie würden Sie das Ausführungsrecht für dieses Programm auf einen bestimmten User („operator“) beschränken?

Aufgabe 5.5: (30 Punkte)
Implementieren Sie in der Programmiersprache C ein Programm für Windows, welches die Security Descriptor Informationen für einen Datei/ein Verzeichnis anzeigen kann! Verwenden Sie die in der Übung vorgestellten Windows-API Funktionen!

Es sollen mindestens die folgenden Elemente angezeigt werden:

- Besitzer und Gruppe des Besitzers der Datei
- Liste der ACEs mit Angabe der Rechte-Maske, der Rechte und der entsprechenden Benutzergruppe / des entsprechenden Benutzers.

Beispiel:

```
> sdviewer test.txt
Security Descriptor for test.txt
        owner = HPI\Max.Mustermann
        group = HPI\Domain Users

        ACEs:
          0: - mask = 0xf01ff
                 DELETE
                 READ_CONTROL
                 WRITE_DAC
                 WRITE_OWNER
                 STANDARD_RIGHTS_REQUIRED
               BUILTIN\Administrators
          1: + mask = 0x1f01ff
                 DELETE
                 READ_CONTROL
                 WRITE_DAC
[…]
          4: + mask = 0x1200a9
                 READ_CONTROL
                 SYNCHRONIZE
               BUILTIN\Users
```

Erläutern Sie Ihrem Tutor Ihre Implementierung und führen Sie Ihr Programm vor! Erklären Sie mit Hilfe der Programmausgaben beispielhaft, wie die Überprüfung der Zugriffsrechte in Windows erfolgt!

Erstellen Sie ein Makefile für **`nmake`** mit dem Ziel **`sdviewer.exe`**. Verpacken Sie alle Source-Code Dateien (inklusive Makefile) in eine ZIP-Datei (**`blatt5.zip`**). Diese ZIP-Datei muss mindestens 24 Stunden vor dem Tutoriumstermin in das Abgabesystem eingestellt werden.

Aufgabe 5.6*: (20 Punkte)
Implementieren Sie in der Programmiersprache C einen einfachen Echo-Dienst für Windows! Erstellen Sie ein Client- und einen Server-Programm, die mit Hilfe von TCP-Sockets kommunizieren. Es soll die Windows Socket API verwendet werden.

Der Client soll Zeilen von der Standardeingabe lesen und als TCP-Paket an den Server senden; der Server wandelt alle Buchstaben der Zeichenkette in Großbuchstaben um und sendet das Ergebnis zurück an den Client.

Erläutern Sie Ihrem Tutor Ihre Implementierung und führen Sie Ihr Programm vor!

Erstellen Sie ein Makefile für **`nmake`** mit den Zielen **`echoclient.exe`** und **`echoserver.exe`**. Verpacken Sie alle Source-Code Dateien (inklusive Makefile) in eine ZIP-Datei (**`blatt5bonus.zip`**). Diese ZIP-Datei muss mindestens 24 Stunden vor dem Tutoriumstermin in das Abgabesystem eingestellt werden.