Hochschule Augsburg
Prof. Dr. Hubert Högl
E-mail: Hubert.Hoegl@hs-augsburg.de
WWW: http://hhoegl.informatik.hs-augsburg.de/hhwiki/SysProg

Vorbereitung zur Klausur in Systemnaher Programmierung

Übungsblatt 10

- Worin unterscheiden sich Programmiersprachen die sich zur maschinennahen (= systemnahen) Programmierung eignen von Sprachen, die man "high-level" Sprachen nennt? (Übung 1, A2)
- Anwendungsbereiche von systemnahen Programmiersprachen (Übung 1, A3)
- Die Register der x86 CPU
- Welche Adressierungsarten gibt es? (Bartlett, Kap 2)
- Datensätze ("records")
- Was ist virtueller Speicher?
- Verständnisfragen zu prog-3-1 und prog-3-2 (Übung 2, 1. und 2. Aufgabe)
- Was sind und wie funktionieren Systemaufrufe?

  Wie unterscheiden sie sich von Funktionsaufrufen?

  Beispiele: exit(), read(), write(), lseek(), ...

  Wie wird der Teil des Betriebssystems genannt, in den die Systemaufrufe springen?
- Elementares GDB Verständnis (Übung 2, 2. Aufgabe, gdb.html)
- Wozu braucht man die `-g` Option beim C Compiler?
- Struktur einer ausführbaren Datei im Speicher (Sektionen, Stack, Parameter, ...)
- Wie wird ein Programm geladen und ausgeführt?
    - Starten über Shell bzw. exec()
    - Kopieren in den Hauptspeicher
    - Einsprung an Startadresse
    - Systemaufrufe
- Minimum/Maximum Suche (Übung 3)
    - Varianten zur Angabe der Array-Länge
- Wozu braucht man den Stack und wie funktioniert er?

- Umwandlung Zahl in ASCII
- Stack Frame (Übung 4, Aufgabe 2)
- C Aufrufkonvention (Übung 4, Aufgabe 3)
- Alles über Funktionen (Übung 4, Aufgabe 4)
    - Sehen Sie sich nochmal den Kurztest an (im Repository an der Stelle `tests/timespow/timespow.s`).
    - Schreiben Sie `maximum(ptr_to_list)` (Übung 4, Aufgabe 5)
- Textdateien lesen und schreiben (Übung 5)
    - Umwandlung Gross-/Kleinschreibung
- Was sind "robuste" Programme? (Bartlett, Kap. 7)
- Kommandozeilenargumente (Übung 5)
- Programme aus Teilen zusammenbauen
    - Quelltext inkludieren (Nachteil?)
    - Objektdateien (.o)
    - "Linken" von Objektdateien
- Bibliotheken
    - Statisch
    - Dynamisch (Übung 7)
    - Beispiel: Standard C Bibliothek (libc.a)
    - Ist `printf()` ein Systemaufruf oder eine Bibliotheksfunktion?
- Wie funktioniert die dynamische Speicherverwaltung ("heap"), Bartlett Kap. 9, Übung 8.
    - Wieso braucht man sie?
    - Wie funktioniert sie prinzipiell?
    - Systemprogrammiersprachen wie C verwalten die Freigabe manuell, warum?
    - Echte high-level Sprachen, z.B. Java, C#, Python, verwalten die Freigabe automatisch, warum?
- Wie schreibt man eine Zahl auf ein Display. Der Algorithums soll hier im Vordergrund stehen, gerne in C oder Java nachprogrammieren (Bartlett Kap. 10).
- Schreiben Sie ein kleines Programm in Assembler oder C, das zeigt, welche *Endianness* Ihr Rechner hat.
- Wie funktioniert die in Übung 9 geschilderte Einbruchstechnik?
- Was sind lokale und was sind globale Optimierungstechniken (Bartlett Kap. 12).
- Was bedeutet die `-p` Option beim C Compiler?
- Gelegentlich habe ich in die Klausuren ein klein wenig C eingestreut. Siehe die vergangenen Klausuren:

  http://elk.informatik.hs-augsburg.de/hhweb/sysprog/Klausuren/