Technische Universität Berlin Fachgebiet Komplexe und Verteilte IT-Systeme Sommersemester 2018	**Aufgabenblatt 1** zu – Systemprogrammierung – Prof. Dr. Odej Kao, Prof. Dr. Jan Nordholz
Abgabetermin:	[1] – 13.05.2018 23:55 Uhr [2] – 13.05.2018 23:55 Uhr

Aufgabe 1.1: Virtueller Adressraum(1 Punkt) (Theorie[1])

a) Skizzieren Sie die Aufteilung des virtuellen Adressraums. (0.5 Punkte)

b) In welchen Segmenten werden die folgenden Daten gespeichert? (In dieser Veranstaltung wird immer von der Programmiersprache C ausgegangen.) (0.5 Punkte)

   (a) Initialisierte Globale Variablen
   (b) Programmcode
   (c) Initialisierte lokale static Variablen
   (d) Initialisierte lokale Variablen
   (e) Funktionsparameter

Aufgabe 1.2: Betriebssysteme(1 Punkt) (Theorie[1])

a) Erklären Sie die Funktionsweise von Interrupts in ca. zwei Sätzen und geben Sie zwei Beispiele für das Auftreten eines Interrupts an. (0.5 Punkte)

b) Begründen Sie die Notwendigkeit von Systemaufrufen (syscalls) in wenigen Sätzen. (0.5 Punkte)

Aufgabe 1.3: Einführung in C (Tafelübung)

Schreiben Sie ein Programm in C, welches die Fakultät einer Zahl n berechnet und ausgibt. Welche Stufen mit welchen Zwischenergebnissen werden durchlaufen, um aus dem Code ein ausführbares Programm zu erzeugen? Wie würde ein entsprechender, beispielhafter GCC-Aufruf lauten?

Aufgabe 1.4: Von-Neumann-Architektur (Tafelübung)

Die am weitesten verbreitete Rechnerarchitektur wurde nach John von Neumann benannt.

a) Beschreiben Sie die Funktionen der vier grundlegenden Komponenten der Von-Neumann-Architektur:

   - CPU
   - Speicher
   - Ein-/Ausgabegeräte
   - Gemeinsamer Bus

b) Wie wird in einem Rechner auf Basis dieser Architektur ein Programm prinzipiell abgearbeitet (Takte)?

c) Nennen sie zwei Nachteile gegenüber einer parallelen Architektur wie der Harvard-Architektur. Gehen Sie dabei auch auf den *Von-Neumann-Flaschenhals* ein.

Aufgabe 1.5: Stack (3 Punkte) (Praxis^2)

In dieser Aufgabe soll auf Basis einer einfach verketteten Liste ein Stack implementiert werden. Hierzu soll die Datenstruktur und deren Elemente als structs abgebildet werden und die gängigen Funktionen *push*, *peek* und *pop* implementiert werden. Des weiteren sollen die folgenden Funktionen implementiert werden:

a) *stack_new* Um einen neuen leeren Stack zu erzeugen.

b) *s_elem_new* Um ein neues Stack-Element zu erzeugen.

c) *stack_free* Um einen Stack und alle Elemente auf ihm zu löschen und den Speicher frei zu geben.

d) *stack_size* Für die Ausgabe der aktuellen Stackgröße.

e) *stack_print* Für die Ausgabe eines gesamten Stacks auf der Standardausgabe (Auf Formatierung achten!).

Nähere Informationen zu diesen Funktionen finden Sie in der vorgegebenen stack.h.

Die folgenden Fragen können Sie zur Orientierung nutzen.

- Was ist ein struct in C und wie wird es genutzt?
- Wie funktioniert die dynamische Speicherverwaltung über malloc() und free()?
- Was sind Pointer und wie werden sie genutzt?
- Was ist eine einfach verkettete Liste und wie kann diese in C implementiert werden?

Hinweise:

- **Vorgaben:** Bitte halten Sie sich bei der Programmierung immer an die Vorgaben. Eine Missachtung kann zu Punktabzug führen.
- **Makefile:** Bitte verwenden Sie für diese Aufgabe das Makefile aus der Vorgabe.
- **Dynamischer Speicher:** Um zu evaluieren, ob nach dem Aufruf von *pqueue_free()* der gesamte von der Queue allokierte Speicher wieder freigegeben wurde, empfiehlt sich das Kommandozeilen Werkzeug **valgrind**[1] (unter linux über das Packet valgrind). **Memory leaks führen zu Punktabzug.**

[1] http://valgrind.org/