

Übungen zu Systemprogrammierung 1 (SP1)

Ü1-2 – Speicherverwaltung

Jens Schedel, Christoph Erhardt, Jürgen Kleinöder

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen-Nürnberg

WS 2014 – 13. bis 17. Oktober 2014

https://www4.cs.fau.de/Lehre/WS14/V_SP1



Agenda

- 2.1 Verkettete Listen
- 2.2 Übersetzen von Programmen
- 2.3 Portable Programme
- 2.4 Anforderungen an Abgaben
- 2.5 Gelerntes anwenden



Agenda

2.1 Verkettete Listen

2.2 Übersetzen von Programmen

2.3 Portable Programme

2.4 Anforderungen an Abgaben

2.5 Gelerntes anwenden



Verkettete Liste

- Wann setzt man eine verkettete Liste ein?
- Anforderungsanalyse für verkettete Liste
 - Wieviele Listenelemente gibt es maximal?
 - Welche Lebensdauer muss ein Listenelement besitzen?
 - In welchem Kontext muss ein Listenelement sichtbar sein?
- Wir brauchen einen Mechanismus, mit dem Listenelemente
 - in a-priori nicht bekannter Anzahl
 - zur Laufzeit des Programmes erzeugt und zerstört werden können



Dynamische Speicherverwaltung

- Rückblick: Aufgabe 10.1 (Menschenkette) in AuD SS 2012

```
WaitingHuman somebody = new WaitingHuman("Mrs. Somebody");  
WaitingHuman nobody = new WaitingHuman("Mr. Nobody");  
  
somebody.add(nobody);
```

- In Java: Neues Listenelement wird mit Hilfe von `new` instanziiert
 - Reservieren eines Speicherbereiches für das Objekt
 - Initialisieren des Objektes durch Ausführen des Konstruktors

- In C: Anlegen eines Listenelementes mittels `malloc(3)`

```
struct listelement *newElement;  
newElement = malloc( sizeof(struct listelement) );  
if( newElement == NULL ) {  
    // Fehlerbehandlung  
}
```

- Zurückgegebener Speicher hat undefinierten/zufälligen Wert
- Initialisierung muss per Hand erfolgen



Dynamische Speicherverwaltung

- Explizite Initialisierung mit definiertem Wert: `memset(3)`

```
memset(newElement, 0, sizeof(struct listelement));
```

- Mit 0 vorinitialisierter Speicher kann mit `calloc(3)` angefordert werden

```
struct listelement *newElement;  
newElement = calloc( 1, sizeof(struct listelement) );  
if ( newElement == NULL ) { /* Fehler */ }
```

- Im Gegensatz zu Java gibt es in C keinen Garbage-Collection-Mechanismus
 - Speicherbereich muss von Hand mittels `free(3)` freigegeben werden
 - Nur Speicher, der mit einer der Funktionen `malloc(3)`, `calloc(3)` oder `realloc(3)` angefordert wurde, darf mit `free(3)` freigegeben werden!
 - Zugriff auf freigegebenen Speicherbereich ist undefiniert



Agenda

- 2.1 Verkettete Listen
- 2.2 Übersetzen von Programmen
- 2.3 Portable Programme
- 2.4 Anforderungen an Abgaben
- 2.5 Gelerntes anwenden



Compiler und Optionen

- Übersetzen einer Quelldatei mit gcc

```
> gcc -o test test.c
```

- Zur Erinnerung: Starten der ausführbaren Datei test mit ./test

- Verhalten des gcc kann durch Optionen beeinflusst werden

- -g: Erzeugt Debug-Symbole in der ausführbaren Datei
- -c: Übersetzt Quellcode in Maschinencode, erzeugt aber kein ausführbares Programm
- -Wall: aktiviert weitere Warnungen, die auf mögliche Programmierfehler hinweisen
- -Werror: gcc behandelt Warnungen wie Fehler



Gängige Compiler-Warnungen

- *implicit declaration of function 'printf'*

- bei Bibliotheksfunktionen fehlt entsprechendes `#include`

- entsprechende Manual-Page gibt Auskunft über den Namen der nötigen Headerdateien

```
> man 3 printf
SYNOPSIS
    #include <stdio.h>

    int printf(const char *format, ...);
```

- bei einer eigenen Funktionen fehlt die Forward-Deklaration

- *control reaches end of non-void function*

- in der Funktion, die einen Wert zurückliefern soll, fehlt an einem Austrittspfad eine passende `return`-Anweisung



Agenda

- 2.1 Verkettete Listen
- 2.2 Übersetzen von Programmen
- 2.3 Portable Programme
- 2.4 Anforderungen an Abgaben
- 2.5 Gelerntes anwenden



Portable Programme

- Entwicklung portabler Programme durch Verwendung definierter Schnittstellen

ANSI C99

- Normierung des Sprachumfangs der Programmiersprache C
- Standard-Bibliotheksfunktionen, z. B. `printf`, `malloc`

Single UNIX Specification, Version 4 (SUSv4)

- Standardisierung der Betriebssystemschnittstelle
- Wird von verschiedenen Betriebssystemen implementiert:
 - Solaris, HP-UX, AIX (*SUSv3*)
 - Mac OS X (*SUSv3*)
 - ... und natürlich Linux (*SUSv4, aber nicht offiziell zertifiziert*)



... und was ist mit Windows?

ANSI C99

- Von Microsoft Visual C/C++ nicht unterstützt :-)
- GCC läuft auch unter Windows :-)

Single UNIX Specification, Version 4 (SUSv4)

- Von Microsoft Windows nicht unterstützt :-)
- UNIX-Kompatibilitätsschicht für Windows: Cygwin (<https://cygwin.com/>)
 - ... ist aber eher frickelig :-|

- Ihr wollt eure SP-Programme unter Linux entwickeln. Wirklich.



Agenda

- 2.1 Verkettete Listen
- 2.2 Übersetzen von Programmen
- 2.3 Portable Programme
- 2.4 Anforderungen an Abgaben
- 2.5 Gelerntes anwenden



Anforderungen an abgegebene Lösungen

- C-Sprachumfang konform zu ANSI C99
- Betriebssystemschnittstelle konform zu SUSv4
- **warnings-** und **fehlerfrei** im CIP-Pool mit folgenden gcc-Optionen übersetzen

```
-std=c99 -pedantic -D_XOPEN_SOURCE=700 -Wall -Werror
```

- `-std=c99 -pedantic` erlauben nur ANSI-C99-konformen C-Quellcode
- `-D_XOPEN_SOURCE=700` erlaubt nur SUSv4-konforme Betriebssystemaufrufe
- einzelne Aufgaben können hiervon abweichen, dies wird in der Aufgabenstellung entsprechend vermerkt



Agenda

- 2.1 Verkettete Listen
- 2.2 Übersetzen von Programmen
- 2.3 Portable Programme
- 2.4 Anforderungen an Abgaben
- 2.5 Gelerntes anwenden



„Aufgabenstellung“

- Optional: Programm schreiben, welches „Hallo Welt!“ ausgibt
- Sieb des Eratosthenes implementieren
 - Erstes Argument gibt an, bis zu welcher Zahl geprüft werden soll
- Programme übersetzen

