Anhang III:

Arbeiten am CIP–Pool in C++

Der CIP–Pool

Im Rahmen der Übungen zur Vorlesung ”Algorithmen und Programmiertechniken” bieten wir Ihnen die Möglichkeit, Übungsaufgaben, bei denen programmiert werden soll, an den Rechnern des CIP–Pools des Lehrstuhls für Allgemeine Elektrotechnik und Datenverarbeitungssysteme zu lösen. Um einen reibungslosen Betrieb der Übungen zu erreichen, bitten wir Sie, die folgenden Punkte zu beachten:

- Die Übungen werden im CIP–Pool von uns montangs zwischen 14:00 und 16:00 Uhr betreut.
- Sie können die Aufgaben auch zu den normalen Öffnungszeiten des CIP–Pools bearbeiten. Dann steht allerdings niemand zu Verfügung, um übungsspezifische Fragen zu beantworten. Bitte stellen Sie solche Fragen nur während der Beratungszeit am Mittwoch oder in Ihrer Übungsgruppe.

Die Rechner im CIP–Pool

Im CIP–Pool stehen etwa 80 PCs zur Verfügung. Alle Geräte bieten die gleichen Voraussetzungen für das Programmieren, da sie vernetzt auf den selben Server zugreifen. Die Rechner sind so konfiguriert, daß nach dem Einschalten ein Auswahlmenü erscheint, das die Wahl zwischen drei Betriebssystemen anbietet: DOS, Windows NT und Linux. Falls noch Windows NT läuft muß es erst heruntergefahren werden (mit `Strg+Alt+Entf`, `Herunterfahren`, `Herunterfahren und neu starten`), bei DOS genügt `Strg+Alt+Entf`. Der Rechner bootet und das Auswahlmenu erscheint. Mit den Pfeiltasten (Cursortasten) wird Linux gewählt und danach mit der Eingabetaste (↵) bestätigt.

Kennungen

Für die Übungen können die Kennungen aus dem AI–Praktikum I verwendet werden. Wer keinen solchen Account hat, kann ihn beim Übungsgruppenleiter oder bei unserer CIP–Beratung beantragen.

Anmelden / Einloggen

Nachdem als Betriebssystem Linux ausgewählt wurde, bootet der Rechner und meldet sich mit einem Text–Login, nach einer kurzen Zeit

auch mit einem graphischen Login. Dort wird die Kennung und das Paßwort eingegeben. Ist die Anmeldung erfolgreich, erscheinen auf dem Bildschirmhintergrund einige Buttons und ein Fenster mit einer Kommandozeileneingabe (Shell). Eine Einführung in X, den Fvwm2 oder Unix kann hier nicht gegeben werden, daher hier nur kurz das Wichtigste.

Wenn Sie auf dem Hintergrund die linke, mittlere oder rechte Maustaste drücken bekommen Sie jeweils verschiedene Menus in denen Sie Programme starten, zu anderen Fenster wechseln und Fensterfunktionen aufrufen können.

Als Windowmanager ist der Fvwm2 eingestellt, mit ’Andere Fenstermanager – Fvwm95 starten’ können Sie den Fvwm95 starten, der eine etwas Windows95–ähnlichere Oberfläche bietet. Im graphischen Login können Sie unter ‘Session Type’ auch kde wählen, der moderner daherkommt.

Unter den fvwm–Windowmanagern gibt es einen *Pager* (oben links) mit dem Sie zwischen verschiedenen *virtuellen Bildschirmen* umschalten können (das geht auch über Tastatur mit Shift+Alt+Cursortaste). In kde läßt sich das einstellen, defaultmäßig wird mit Strg–Tab gewechselt.

Abmelden / Ausloggen

Am Ende einer Sitzung führen Sie bitte folgende Schritte aus, damit der Rechner anschließend in einem vernünftigen Zustand ist:

- Verlassen Sie einen ggf. geöffneten Emacs durch die Tastenkombination Strg–x Strg–c. Ggf. fragt der Emacs noch, ob nicht gespeicherte Dateien gespeichert werden sollen, dann schließt sich das Editorfenster.
- Mit Arbeitsmenu – Fenstermananger – Fvwm2 und X beenden wird XFree86 beendet.
- Den Rechner kann am Text–Login mit Strg+Alt+Entf oder im graphischen Login über den Beenden–Button herunterfahren werden.

`man` *befehl*	Anzeigen der Hilfeseite zu einem Befehl.
`ls` *[pfad]*	Anzeigen des Inhalts eines Verzeichnisses.
`cd` *[pfad]*	Wechseln in ein anderes Verzeichnis. Wird *pfad* weggelassen, wird ins Homeverzeichnis gewechselt.
`cp` *datei ziel*	Kopieren einer Datei. *ziel* muß angegeben werden und kann ein neuer Dateiname, der Name einer existierenden Datei oder ein Verzeichnis sein.
`mv` *datei ziel*	Verschieben oder Umbenennen einer Datei.
`rm` *datei*	Löschen einer Datei.
`rmdir` *verzeichnis*	Löschen eines leeren Verzeichnisses.
`mkdir` *verzeichnis*	Anlegen eines Verzeichnisses.
`yppasswd`	Ändern des Passworts.
`emacs`	Starten des Editors Emacs.
`g++` *datei.cpp*	Übersetzen eines Programms *datei.cpp* in eine ausführbare Datei *a.out*.
`g++` *datei.cpp –o datei* oder `make` *datei*	Übersetzen von *datei.cpp* in eine ausführbare Datei *datei*.
`exit` oder `logout`	Ausloggen und Beenden der Sitzung in einer Shell.

Die wichtigsten Kommandos unter Linux

C++ mit Emacs und g++

Wir empfehlen unter Linux als Editor den Emacs und als Compiler den GNU g++ zu verwenden. Die folgenden Schritte geben einen beispielhaften Durchlauf durch eine Programmiersitzung mit dem Emacs und g++ an.

1. Wechseln Sie mit dem *Pager* auf einen freien virtuellen Bildschirm.
2. Starten Sie den Emacs (z.B. mit der linken Maustaste in dem erscheinenden Arbeitsmenu oder über den Button “Tools”).
3. Öffnen Sie mit `Strg-x Strg-f` eine Datei `hello.cc`. Da die Datei noch nicht existiert, wird sie neu erzeugt. (Hinweis: auf die Dauer ist es sinnvoll, mehrere Verzeichnisse für die Aufgaben anzulegen.)
4. Jetzt können Sie einen beliebigen Text eingeben, z.B. folgendes Programm, das ’`Hello, World`’ ausgibt.

```
#include <iostream.h>
int main()
{
   cout << ”Hello, World!” << endl;
   return 0;
}
```

Tip: Benutzen Sie `Tab`, um die Quelltextzeilen einzurücken.

5. Mit `Strg–x Strg–s` wird das eingegebene Programm gespeichert.

6. Das Programm soll nun übersetzt und gestartet werden. Zur Übung wollen wir das zunächst von Hand in der Shell durchführen. Dazu

   (a) Mit dem *Pager* in den virtuellen Bildschirm mit der Shell wechseln oder im Arbeitsmenu eine neue Shell öffnen.

   (b) In der Shell `g++ hello.cc` eingeben. Dadurch wird der eingegebene Quelltext in ein ausführbares Programm (Defaultname `a.out`) übersetzt. Sollten beim Übersetzen Fehler auftreten, so müssen Sie den Programmtext nochmal gründlich untersuchen und diese Fehler beseitigen. Danach muß erneut gespeichert und übersetzt werden.

   (c) War das Übersetzen erfolgreich (der Compiler hat keine Fehlermeldung ausgegeben), so kann das Programm jetzt mit ’`a.out`’ gestartet werden und in der Shell erscheint die Ausgabe

```
Hello, World!
```

C–x C–f	Laden oder Erzeugen einer Datei.
C–x C–s	Speichern einer Datei.
C–g	Abbrechen einer Operation (zurück in den Normalzustand :–).
C–x C–c	Beenden des Emacs.
M–x *befehl*	Eingabe eines Kommandonamens von Hand.
C–h f	Hilfe zu einer Funktion.
C–h k	Hilfe zu einer Tastenkombination.
C–h i	Startet info. Hier gibt es Hilfetexte zum Emacs, dem gcc, der C++–Standardbibliothek usw.
C–space	Start eines Blocks markieren.
C–w	Block löschen.
C–w	Block an Cursorposition einfügen.
C–k	Zeile von Cursorposition bis Ende löschen.
C–s	Suchen nach einem Text.
M–%	Suchen und Ersetzen.

Die wichtigsten Kommandos im Emacs
Gemäß den Emacs–Konventionen steht ‘C–’ für ‘Strg–’ und
‘M–’ für ‘Meta’, auf der Tastatur als ‘Alt–’ oder ‘Esc’.

Weitere Hinweise

Debugging

Der Emacs kann auch den gdb, den Gnu DeBugger, integriert betreiben. Mit M–x gdb wird der gdb–Modus aufgerufen, wozu die Kommandozeile abgefragt wird, etwa gdb Hello. Es öffnet sich ein Emacs–Fenster mit der gdb–Konsole. help zeigt eine Liste der Befehle, hier interessant sind b *funktion* (Programm bei Eintritt in *funktion* unterbrechen), n (Programm bis zur nächsten Zeile ausführen), s (Einzelschritt, auch in Funktionen hinein), c (Programm weiter ausführen), p *ausdruck* (*ausdruck* auswerten und ausgeben). Der Emacs zeigt dabei nach Möglichkeit die passenden Quelltextzeilen an.

Make

Mit make kann die Programmerstellung automatisiert werden. Anleitungen hierzu werden Dateien namens makefile oder Makefile

entnommen. Vieles ‘weiß’ `make` sowieso, aber ein Makefile mit dem Inhalt `CPPFLAGS=-ansi -pedantic -Wall -g` ist hier nützlich, weil es `make` dazu anweist, beim Compilieren von C++–Quelltexten vom Compiler maximal viele Warnungen zu melden und Informationen für den Debugger abzulegen.

EMail

Post läßt sich im CIP–Pool noch am ehesten mit Netscape (über den Button ‘Tools’ zu haben) verschicken und lesen. Der zu nutzende Dienst ist POP3, der Benutzername auf dem Server sollte bereits eingetragen sein und das Paßwort ist die Benutzernummer.

Übungsaufgaben in C++

Hier ein paar Aufgaben, die Sie bearbeiten können, um mit dem System vertraut zu werden.

1. Schreiben Sie ein Programm, das nach den folgenden Daten eines Studenten bzw. einer Studentin fragt, sie einliest und anschließend wieder ausgibt: (Name, Vorname, Matrikelnummer und Geschlecht). Sie können folgendes Programmskelett verwenden

```
#include <iostream.h>
char name[30]; // name speichert 29 Zeichen
               // +abschliessendes Nullbyte
// ...
int main()
{
    // Daten eingeben
    cout << ”Name eingeben: ”;
    cin >> name;
    // ...
    // Daten ausgeben
    cout << ”Name: ” << name << endl;
    // ...
    return 0;
}
```

2. Die Fibonacci–Zahlen sind wie folgt definiert:

$$Fib(1) = 1$$
$$Fib(2) = 1$$
$$\forall_{(i>2)}\ Fib(i) = Fib(i-1) + Fib(i-2)$$

(a) Schreiben Sie ein Programm, daß eine Zahl `N` einliest und die `N`te Fibonacci–Zahl berechnet und ausgibt.

(b) Erweitern Sie das Programm so, daß die ersten `N` Fibonacci–Zahlen berechnet und ausgegeben werden.

3. Schreiben Sie ein Programm, das zwei positive ganze Zahlen einliest und deren größten gemeinsamen Teiler und kleinstes gemeinsames Vielfaches berechnet und ausgibt.

4. Schreiben Sie ein Programm, daß die Werte der Sinusfunktion im Intervall [0, 2π[berechnet. Das Programm soll die Werte in einem Abstand von $\frac{\pi}{25}$ berechnen, also $\sin 0, \sin\frac{\pi}{25}, \sin\frac{2\pi}{25}, \ldots, \sin\frac{49\pi}{25}$.
Um die Funktion am Bildschirm anzuzeigen, speichern sie die berechneten Werte in einer zweidimensionale Feldvariable mit-Komponenten vom Typ CHAR wie folgt ab: Eine Dimension des Feldes entspricht der X–Achse, die andere der Y–Achse. Das gesamte Feld wird mit Leerzeichen initialisiert. Nun soll für jeden berechneten Funktionswert ein Stern in die Komponente des Feldes geschrieben werden, die der (X,Y)–Position des berechneten Wertes am nächsten kommt. Danach wird das Feld Zeichenweise auf dem Bildschirm ausgegeben (am Ende einer Zeile ein `cout << endl;` (Zeilenumbruch) einfügen).
Ein recht guter Wert für die Größe der Y–Dimension des Feldes ist 24, da das ungefähr der Bildschirmgröße entspricht.
Um einen ganzzahligen Wert `i` in einen reellwertigen umzuwandeln können Sie z.B. `float f = float(i)` verwenden. Umgekehrt können Sie mit `i = int(f)` einen `float` in einen `int` wandeln.

Experimentieren Sie ein wenig mit den Parametern und der dargestellten Funktion.