

Wissenschaftliches Programmieren für Ingenieure

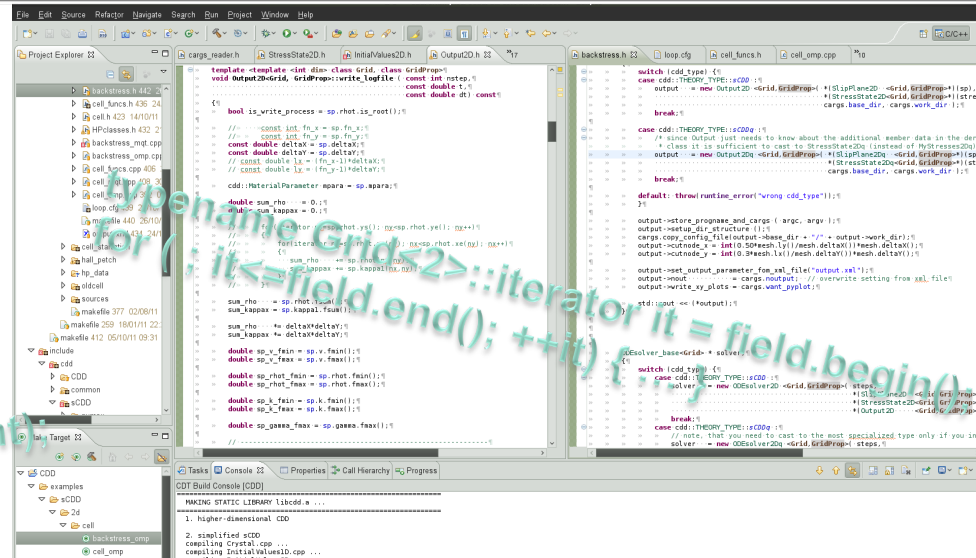
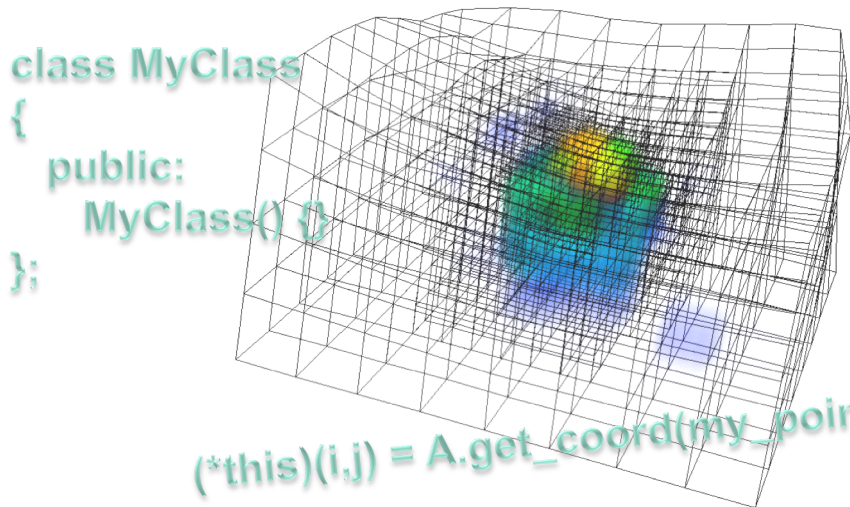
T. Achkar, Dr. D. Weygand

Übung 1:

Einrichten der UNIX Programmierungsumgebung, erste C++ Programme und Datenvisualisierung mit GNUPLOT

Institute for Applied Materials – Computational Materials Science (IAM-CMS)

KIT – The Karlsruhe Institute of Technology – University of the State of Baden-Wuerttemberg, GERMANY




Erste Schritte unter Unix/Linux

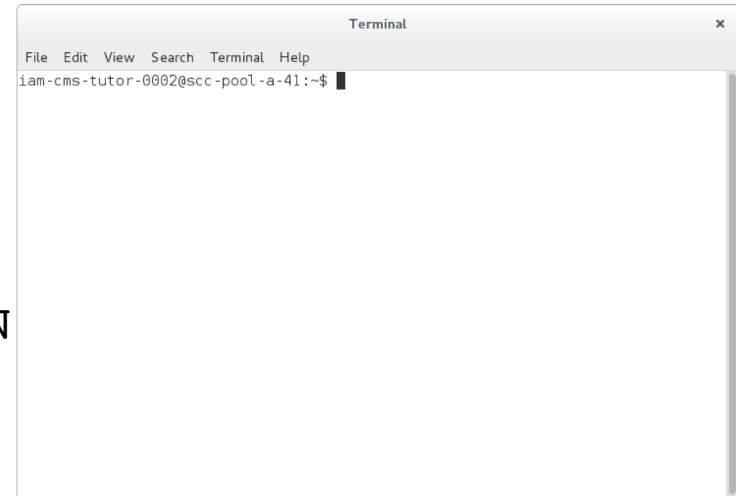
Info

- Beim Starten den Rechners: Linux als Betriebssystem auswählen
- Login mit dem normalen Studentenzugang des SCC/KIT
- Nach dem Login:
 - Terminal öffnen: Details siehe nächste Folie
- Die Konsole ist die Schnittstelle zum Nutzer: Befehle werden auf der sogenannten Kommandozeile eingegeben
 - Die Konsole nimmt Kommando entgegen und wertet diese aus: Kommandozeileninterpreter oder auch in englisch Shell
 - Shell: Schnittstelle zwischen Nutzer und Betriebssystem – es gibt eine Vielzahl von verschiedenen Shells (sh, ksh, bash, csh, zsh,...) wobei wir hier nur auf die bash Shell eingehen werden (sh-artige Shell)
 - Aufgaben der Shell:
 - Interpreter der Kommandos – im Prinzip eine eigene Programmiersprache (Skriptsprache)
 - Manipulation von Daten: siehe Skript und `unix_manual.pdf`

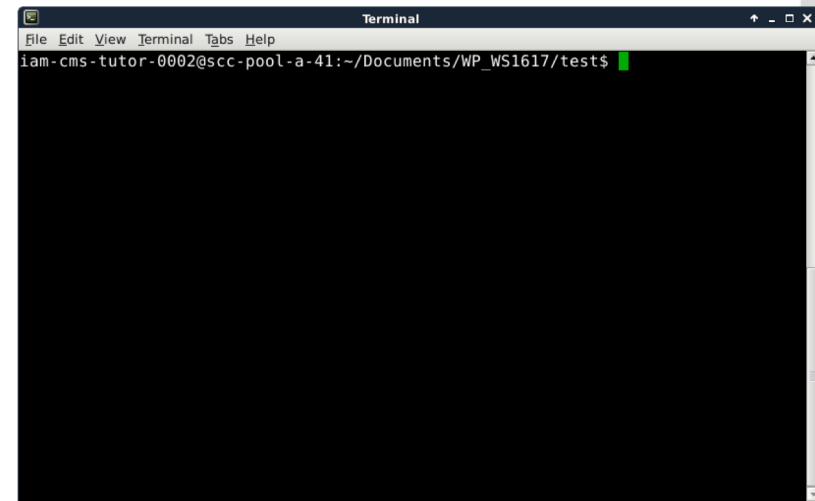
Terminal Varianten der grafischen Oberflächen

Info

- GNOME: terminal
Maus in die linke obere Ecke, dann im Menu das Symbol  auswählen.
Oder in der Suchleiste terminal RETURN eingeben.



- LXDE
Start
 - > System Tools
 - > Xfce Terminal



- XFCE 4
Im Menu links oben:
Applications Menu
 - > Terminal Emulator

Graphische Benutzeroberflächen im Debian System des SCC

Todo

Ziel: Eine Fenster mit einer Kommandozeile soll zugänglich sein!

■ GNOME (Standardumgebung)

- Erlaubt den Zugriff auf die installierten Programme
- Terminal: Kommandozeile
- Editoren: emacs, gedit
- Webbrowser: iceweasel (Firefox Derivat)
- Compiler: g++

■ KDE (ebenfalls installiert)

- konsole: Terminalfenster mit Kommandozeile
- Editoren: emacs, kedit (8tung: kedit nur unter KDE benutzen)
- Compiler: g++

1. UNIX Umgebung

Todo

Herunterladen der Übungsunterlagen von ILIAS

- Laden Sie die Dateien von ILIAS herunter:
 - Browser aufrufen und im Ilias System einloggen:
 - Dateien herunterladen
- Öffnen Sie eine Terminal (Punkt 1 der vorigen Folie)
- finden Sie die heruntergeladenen Dateien:

Sie können mit `cd Verzeichnisname` durch Ihre Verzeichnisstruktur navigieren.

Mit `ls` können Sie den jeweiligen Inhalt sehen (zu den UNIX Befehlen siehe auch `unix_intro.pdf` in ILIAS).
- Öffnen sie das pdf entweder über das GUI / Filebrowser oder über die Kommandozeile:

Eingabe: `evince name_der_datei.pdf` RETURN

 - *evince* ist ein Programm zum Betrachten von pdf oder postscript basierten Dokumenten (Ersatz für Acrobat Reader)

2.1 Dateien erstellen, Daten plotten unter Linux

Todo

Editor aufrufen

- Um Textdateien zu bearbeiten brauchen Sie einen Texteditor, z.B. `emacs`, `gedit` oder `kate`.
- Sie starten den Editor auf der Konsole durch Eingabe von `emacs &`.

Kommandozeile: Programme unterbrechen, wiederaufnehmen, abbrechen

- Das `&`-Zeichen bringt das Programm in den „Hintergrund“, d.h. Sie können die Kommandozeile weiter benutzen, obwohl `emacs` noch nicht beendet ist.
Ohne das `&`-Zeichen muss `emacs` beendet werden, bevor man auf der Kommandozeile wieder etwas tippen kann.
- **Programme anhalten und in den Hintergrund bringen:**
 - Um ein Programm anzuhalten drücken Sie `Strg + Z`.
 - durch Eingaben von `bg` (“background”) `RETURN` wird das Programm in den Hintergrund geschickt.
- Um laufende Programme auf der Konsole abzuberechnen drücken Sie `Strg + C`.

2.2 Dateien und Verzeichnisse erstellen

Todo

Verzeichnis erstellen, navigieren in Kommandozeile

- Erstellen Sie ein neues Verzeichnis für diese Übung und gehen Sie dort hinein:

```
mkdir Uebung1      RETURN
```

und

```
cd Uebung1        RETURN
```

- Verschieben Sie die von ILIAS heruntergeladenen Dateien in den Ordner Uebung1 (Kommando `mv`, „move“)

- Erste eigene Datei erstellen: Schritte

Datei erstellen und speichern

- Erstellen Sie eine neue Datei im Editor Ihrer Wahl
- Schreiben Sie etwas hinein
- Speichern Sie den Inhalt ab
- lassen Sie sich mit `ls -l` Informationen über die Datei anzeigen.

Dateiinformatoren in Kommandozeile anzeigen

Welche Informationen erhalten Sie?

2.3 Daten plotten unter Linux

Todo

Vorhandene Daten mit Gnuplot darstellen

- Laden Sie die Datei `wave.dat` , in Ihren Editor;

- Datei befindet sich auf ILIAS

- Programm: gnuplot

- erlaubt die Darstellung von Daten in zwei und drei Dimensionen

- um sich diese Datei grafisch darstellen zu lassen, gebe Sie auf der Konsole `gnuplot` ein.

- Jetzt befinden Sie sich auf der Kommandozeile des Programms `gnuplot`:
Der Befehl

```
plot 'wave.dat' using 1:2 with linespoints RETURN
(Kurzform: p 'wave.dat' u 1:2 w lp)
```

plottet die Daten der Spalte 1 gegen die Spalte 2 der Datei.

- Um `gnuplot` zu verlassen geben Sie `q` + `RETURN` ein.
(siehe ILIAS/gnuplot_manual)

```
# Daten fuer einer periodischen Messung:
# Wiederholung alle 4 Messungen
# Zeit in [s]      Auslenkung [mm]
0.00              1.0
0.25              25.0
0.50              4.0
0.75              -20.0
1.00              1.0
1.25              25.0
1.50              4.0
1.75              -20.0
2.00              1.0
2.25              25.0
2.50              4.0
2.75              -20.0
```

Dateiname: `wave.dat`

2.3 Daten plotten unter Linux

Todo

Funktion mit Gnuplot darstellen

- Neben vorhandenen Daten, lassen sich mit Gnuplot auch Funktionen darstellen
- Gnuplot öffnen durch `gnuplot` RETURN
- Funktion definieren
`gnuplot> f(x) = sin(x)`
- Funktion darstellen
`gnuplot> pl f(x) w lp`
- Wertebereich ändern (y-Achse analog mit `set yrange [min:max]`)
`gnuplot> set xrange [0:2*pi]`
`gnuplot> replot`
- Variablen in Gnuplot, hilfreich zum Abschätzen von funktionalen Zusammenhängen
`gnuplot> a=3; b=0.5; g(x)= a*x+b`
`gnuplot> pl g(x) w lp`

3.1 Ein erstes C++ Programm

Todo

Ziel: Datei öffnen => Quellcode schreiben => übersetzen => ausführen

■ Vorgehensweise:

1. `gedit` oder `emacs` starten (siehe Kurzeinführung zu `emacs` am Ende)
2. Neues Dokument öffnen und z.B. unter Dateinamen `hello_world.cpp` speichern
3. Programm schreiben (siehe Vorlesung), das „hello KIT“ auf dem Bildschirm ausgibt und speichern
4. Programm kompilieren mit `g++ hello_world.cpp -o hello_world`
5. Zum Starten des Programms `hello_world` auf der Konsole `./hello_world` RETURN eingeben

Tipp: bringen Sie den Editor in den Hintergrund (background), dann können Sie in der selben Konsole weiter arbeiten und kompilieren!

Wie ging das nochmal? (Seite 6)

Info

Gängige Endungen (Suffix) für C++ Quellcode sind `.cxx` `.cpp` `.c++`

3.2 Ein zweites C++ Programm

Todo

■ Vorgehensweise:

1. kate oder emacs starten
2. Neues Dokument öffnen und unter Dateinamen `sinus_plot.cxx` speichern
3. Programm schreiben (siehe nächste Folie) und speichern
4. Programm kompilieren mit `g++ sinus_plot.cxx -o plot_me`
5. Zum Starten des Programms `plot_me` auf der Konsole `./plot_me` eingeben

Tipp: bringen Sie den Editor in den Hintergrund (background), dann können Sie in der selben Konsole weiter arbeiten und kompilieren!

3.2 Ein zweites C++ Programm

Todo

- Ziel des Programms: Ausgabe von $\sin(x_i)$ an den Stützstellen x_i
- Bedienung:

Der Benutzer soll den Bereich angeben, in dem $\sin(x)$ evaluiert wird.

Start ist bei 0, Ende ist $hw \cdot \pi$, wobei die Variable hw eine Benutzereingabe ist.

Es soll eingegeben werden können, an wie vielen (äquidistanten) Stellen die Funktion berechnet wird.

```

#include <iostream>
#include <cmath>
                                                    sinus_plot.cxx

using namespace std;

int main(){
    double hw;
    int nsteps;
    const double PI = 3.14159;

    cout<<"Plotten einer Sinuskurve f(x) = sin(x)"<<endl;
    cout<<"-----";
    cout<< "Anzahl der sinus Halbwellen (= Vielfaches von PI): ";
    cin >> hw;

    cout <<" Anzahl der Stuetzstellen: ";
    cin >> nsteps;
    const double maxx = hw * PI;
    const double deltax = maxx/nsteps;

    cout<<" Der x-Bereich liegt jetzt zwischen 0 und " <<maxx<<endl
        << " Diskretisierung durch "<< nsteps<< " Stuetzstellen "
        <<endl<<endl
        << " x      f(x) " <<endl;

    double x = 0.0;

    for ( int i = 0; i< nsteps; i++){
        const double f = sin(x);
        cerr << x<< " " << f << endl;
        x = x + deltax;
    }

    return 0;
}
    
```

3.2 Ein zweites C++ Programm

Todo

Programmanforderungen

- Ausgabe der Benutzeranweisungen auf Standardausgabe (`cout`)
- Ausgabe der berechneten Funktionswerte in Fehlerkanal (`cerr`)
- Test der vom Benutzer/der Benutzerin gewählten Werte auf Plausibilität
- Test der gewählten Stützstellen bezüglich Darstellbarkeit

3.2 Ein zweites C++ Programm

Todo

1. Programm eingeben und kompilieren
2. Leistet das Programm das geforderte?
3. Programm um Tests für Eingabe ergänzen

hw > 0

z.B.

```
if (a>=b){  
    // code fuer a>=b  
}
```

oder

```
if (a>=b){  
    // code fuer a>=b  
} else {  
    // code fuer a<b  
}
```

4. Ausgabe des Programms umleiten (siehe nächste Folie) und anzeigen der erstellten Datei.
5. Hinweis: Gnuplot interpretiert Zeilen mit # am Beginn als Kommentar

4. Shell: Umleitung der Ein- und Ausgabe

Todo

Umleitung der Ausgabe am Beispiel üben

Umleiten von Ausgaben: mit dem `>` Zeichen wird die Standardausgabe in einen anderen Kanal umgelenkt

Beispiel: `./plot_me > sinus.dat`

lenkt die Ausgabe von `plot_me` in die Datei mit dem Namen `sinus.dat` um

- `cout`: schreibt in die Standardausgabe; Umleitung durch `>`
- `cerr`: schreibt in die Standardfehlerausgabe; Umleitung durch `2>`

Umleiten von Eingabe: die Eingaben, die ein Programm auf der Kommandozeile erwartet, können auch in einer Datei gespeichert werden: `start.dat`

und mit `<` Zeichen aus der Datei ausgelesen werden

Beispiel: `./plot_me<start.dat`

Oder beides umgelenkt

Beispiel `./plot_me<start.dat>sinus.dat`

Shell: einige Kommandos / Steuerbefehle

Info

- Navigieren in der Zeile (C steht für [Strg] Taste, M für [Alt] Taste)
 - C-a bzw. C-e Anfang/Ende der Zeile
 - C-f bzw. C-b vorwärts/rückwärts in Zeile (oder Pfeiltasten der Tastatur)
 - C-k bzw. C-y löschen/einfügen
- TAB-Taste : automatisches Ergänzen (solange eindeutig)
 - Beispiel: Kommando `ls` danach TAB-Taste einmal oder wenn nichts passiert zweimal drücken
 - Was passiert?
 - Was passiert, wenn ich einen Buchstaben eingebe, der in der durch `ls` angezeigten Liste als Anfangsbuchstabe vorkommt?
- Wiederholen von Kommandos: `history`
 - `history` (Kommando) zeigt die bisher verwendeten Kommandos an.
 - Systematisches verwenden von schon mal getippten Kommandos:
 - CTRL-r / CTRL-s Rück-/Vorwärtssuche in Kommandohistorie
 - Pfeil-Taste hoch/runter: rückwärts/vorwärts bewegen in Kommandos

Kurzeinführung für emacs (1/2)

Info

- Aufruf in Terminal/Kommandozeile durch: `emacs` RETURN
- Grafische Benutzeroberflächen:
 - selbsterklärend
- Kommandos: (C steht für [Strg] Taste, M für [Alt] Taste)
In den Menüpunkten des Editors werden oft die entsprechenden Tastenkombination angegeben:
 - C-x C-f (zum Öffnen einer Datei, bzw. zum Neuerstellen einer Datei, wenn sie noch nicht existiert).
([Strg] gedrückt halten; hintereinander x und f drücken)
 - C-x C-s :
Speichern der Datei.
 - C-x C-w
Speichern der Datei unter neuem Namen.
 - C-x C-c
Beenden von emacs

Kurzeinführung für emacs (2/2)

Info

- [Bereich markieren] (zum Beispiel mit der Maus)
C-w
Schneidet markierten Bereich aus und speichert ihn in eine Zwischenablage
- C-y
Fügt die Zwischenablage ins Dokument ein (*yank*).
- C-g
Wenn Sie in der Kommandozeile “hängen” geblieben sind: bricht Kommandos von emacs ab.
- **Copy’n Paste:**
Bereich markieren, dann A-w, danach an neue Stelle gehen und zum Einfügen C-y

- Oder alles mit den Menus/Maus erledigen.....