Übung, 16.05.2019
Ziel: Linux Grundlagen und erste Fortran Programme

1 Linux-Grundlagen

1.1 "Konsole"

Die Konsole stellt die Hauptebene der Interaktion mit dem Unix-System dar. Dort können Sie die Befehle ausführen, Programme/Dateien editieren, Programme übersetzen und ausführen. Eine Konsole bzw. Terminal können Sie nach dem Anmelden am System öffnen. Der Name der Konsole/des Terminals hängt von der gewählten graphischen Oberfläche des Linux Systems ab. (KDE: konsole, terminal).

Eine ausführliche online Dokumentation finden Sie unter `https://wiki.ubuntuusers.de/Terminal/`. Die Beschreibung wurde von Canoncial für Ubuntu Linux Distribution geschrieben, ist aber weitgehend allgemeingültig.

Laden Sie aus ILIAS das Übungsarchiv `archiv1.tar` herunter. Dieses können Sie mit der Befehlsfolge `tar -xvf archive1.tar` entpacken. Im Dokument `unix_intro.htm` finden Sie eine Kurzübersicht über wichtige Linux-Befehle, die auf der Konsole verwendet werden können. [1]

1.2 Shell

Öffnen Sie ein "Terminal", das Ihnen eine Kommandozeile zur Verfügung stellt. Die Kommandozeile wird über die "shell command language" (Endung von Scripts `*.sh`) gesteuert. Eine "shell" ist eine Implementierung der Sprache, die Ihre Eingaben interpretiert und ausführt und somit für Sie die Schnittstelle zum Unix/Linux System darstellt. Eine Standardshell ist `bash`, die für uns ausreichend sein wird.

Befehlsübersicht z.B. `https://wiki.ubuntuusers.de/Shell/Befehlsübersicht/`

mkdir Verzeichnis anlegen: `mkdir Uebung01` legt das Verzeichnis mit dem Namen Uebung01 an.

[1] [*] bezeichnen Tasten: [ENTER] die Enter-Taste /Zeilenumbruch

cd Wechseln in ein Verzeichnis mit dem Namen Uebung01: `cd Uebung01`

cp Kopieren einer Datei: `cp Quelldateiname [Zielordner]/[Dateiname]`, wobei Zielordner bzw. Dateiname optional sind (eine der beiden Angaben kann weggelassen werden.

.. : Pfadangabe "..": eine Ebene zurückgehen mit `cd ..` (8tung Leerzeichen zwischen `cd` und `...`

\- : Pfadangabe "-": zurück in das vorherige Verzeichnis mit `cd -`.

ls Auflisten aller Dateien/Verzeichnisse im aktuellen Verzeichnis

mv Bewegen eine Datei im Verzeichnisbaum:`cd Datei.txt ../` (Pfadangabe siehe oben)
Umbenenne einer Datei: `mv Datei.txt NeuerName.txt`

Die Shellumgebung eignet sich auch hervorragend, um Aufgaben zu automatisieren; Eine Einführung in "bash" Shell finden Sie unter und
`https://wiki.ubuntuusers.de/Shell/Bash-Skripting-Guide_für_Anfänger/` oder anderen Internetseiten.

Beispiel: Zahlenfolge von 1 bis 10 in 2 er Schritten:
`echo $(seq 1 2 10)`
`seq` bash Kommando für Zahlenreihe: Startwert Schrittweite Endwert

Beispiel: Anlegen von Verzeichnissen `test1` bis `test10` mit Hilfe einer `for`-Schleife. Kommandos werden durch Strichpunkt getrennt. Der mehrfach ausgeführte Code steht zwischen `do` und `;done`.

```
for i in $(seq 1 10); do mkdir test$i; done
```

Beispiel: Serie von Ganzzahlenpaaren $(i, i * i)$ (Parabel) auf der Konsole ausgeben:
`for i in $(seq 1 20); do echo $i $(($i*$i)); done`
Siehe auch Arithmetik (bash Referenz) zur Erklärung der `$(($i*$i))` Syntax.

1.3 Text-Editor

Sie benötigen einen Texteditor, um Textdateien zu bearbeiten. Ein Standardeditor ist z.B. *emacs*. Viele weitere sind auf dem System installiert (gedit (GNOME-Umgebung), Kedit (KDE-Umgebung),....).

ZUR VERMEIDUNG UNNÖTIGER PROBLEME VERWENDEN SIE BITTE KEINE EDITOREN VON KDE UNTER GNOME UND UMGEKEHRT!!

Sie starten den Editor auf der Konsole durch Eingabe von `emacs & [ENTER]`.

Das `&`-Zeichen bringt das Programm in den Hintergrund, d.h. Sie können die Kommandozeile weiter benutzen, obwohl das Programm `emacs` "läuft". Ohne das `&`-Zeichen

muss `emacs` zunächst beendet werden, bevor man die Konsole weiterverwenden kann.

Es besteht noch eine weitere Möglichkeit ein Programm in den *Hintergrund* zu schicken. Zunächst müssen sie das laufende Programm anhalten: Drücken Sie hierzu `[Strg] + Z`. Nach Eingaben von `bg [ENTER]` (“background”) auf der Konsole wird das Programm dann in den Hintergrund geschickt.

Programmabbruch: drücken Sie `[Strg] + C`, um ein in der Konsole laufendes Programm abzubrechen.

In den Vordergrund holen: tippen sie `fg [Enter]`, um ein im Hintergrund laufendes Programm in den Vordergrund zu holen. Dieses ”blockiert” dann die Konsole.

Kurze Einleitung zum emacs-Editor: klassischer Unix-Editor – und mehr

`http://www.gnu.org/software/emacs/manual/html_mono/emacs.html`

Man die Tastenkombination `C-x C-f` (zum Öffnen einer Datei, bzw. zum Neuerstellen einer Datei, wenn sie noch nicht existiert).

`C-x` bedeutet: Drücken und Halten von `[Strg]` (= `[Ctrl] auf der deutschen Tastatur`) und Drücken von `x`, danach beide Tasten loslassen.

`C-x C-f`
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

[Bereich markieren] `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. 8tung: die Eingabe muss im Gültigkeitsbereich der Kommandozeile sein.

Copy'n Paste:
Bereich markieren, dann `M-w` (M steht für `meta` und ist die `Alt`-Taste), danach an neue Stelle gehen und zum Einfügen `C-y`

`C-Shift-`:
Entspricht `Strg-Z`, also "Rückgängig machen". Emacs ist in dieser Hinsicht nicht eindimensional. Wenn man nach einem "Rückgängig machen" wieder etwas ändert, dann wird die Historie neu. Einfach mal ausprobieren.

`M-g g`
Springe zu einer Zeile in der Datei: emacs erwartet nun die Eingabe einer Zeilennummer + Return.

`M-x query-replace`
Suche "text" und ersetzte diesen durch "Ersatztext"; emacs erwartet nach Aufruf des Kommandos zwei Texteingaben, die jeweils durch Return bestätigt werden müssen. Danach beginnt die Suche. Das Ersetzen von "text" durch "Ersatztext" muss jeweils bestätigt werden.

Oder alles mit den Menüs/Maus erledigen.....

1.4 Aufgaben zu Linux-Grundlagen

Editor: emacs

① Erstellen Sie eine neue Datei, schreiben Sie etwas hinein, danach Abspeichern, sowie Verlassen von `emacs`.

② Machen Sie sich mit der Meldung vertraut, die `emacs` ausgibt, wenn Sie `emacs` verlassen möchten *ohne* gespeichert zu haben. (Die 2 Sterne in der untersten Zeile von emacs zeigen an, dass es ungespeicherte Änderungen gab)

③ Benutzen Sie die Befehle `cd` und `ls` auf der Konsole, um Informationen über die neu erstellte Datei anzeigen zu lassen.

④ Erstellen/Öffnen Sie eine Datei: verwenden Sie die Datei `wave.dat` des ILIAS Servers

```
# 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

⑤ Darstellung mit `gnuplot`: `GnuPlot` ist ein Kommandozeilenorientiertes Plotprogramm auf Unix System. Um die Datei `wave-dat` grafisch darzustellen, gebe Sie auf der Konsole zunächst `gnuplot` ein. Jetzt sind Sie in einer “Plot”-Umgebung. Der Befehl

```
plot 'wave.dat' using 1:2 with linespoints
```

oder die Kurzform: `p 'wave.dat' u 1:2 w lp`

zeichnet (plot) die Datenpunkte der Spalten 1 und 2 gegeneinander auf. Der Dateiname kann in einfachen oder normalen "Hochkommata" eingefasst werden.

8tung: Zeilen in `wave.dat`, die mit # anfangen, werden von `gnuplot` ignoriert (sind Kommentar).

Geben Sie `q + ENTER` ein, um GnuPlot zu verlassen, .

2 Fortran-Einstieg: ein “hands-on approach”

Wir werden uns einige Fortran Grundlagen direkt an Programmbeispielen erarbeiten. Es soll ein Programm erstellt und übersetzt werden. Sie können als Vorübung auch das "Hello World" Programm aus der Vorlesung verwenden, um die Schritte Quellcodeerstellung – Abspeichern – Übersetzen – Ausführen des Programms nachzuvollziehen.

2.1 Vorgehensweise

① Emacs starten: wird für den Quellcode benötigt.

② Neues Dokument öffnen (`C-x C-f`) und Dateinamen `sinus_plot.f90` eingeben (alternative Hello World Programm).

③ Programm schreiben + speichern (`C-x C-S`)

④ Programm übersetzten (kompilieren) mit
`gfortran sinus_plot.f90 -o plot_me`

⑤ Fehler suchen ⇒ gehe zu ④ solange bis keine Bugs mehr im Programm sind (hier sollte alles richtig sein: Schritt entfällt)

⑥ Zum Starten des Programms `plot_me` auf der Konsole `./plot_me` eingeben

2.2 Anwendung: Ausgabe von $sin(x_i)$ an definierbaren Stützstellen x_i

Der Benutzer soll den Bereich zur Auswertung der Funktion $sin(x)$ angeben. Start ist bei 0, Ende ist $hw \cdot \pi$, wobei hw (Halbwelle) eine Benutzereingabe ist.
Es soll eingegeben werden können, an wie vielen (äquidistanten) Stellen die Funktion ausgerechnet wird.

```
!- Kommentare haben  ein '!' am Anfang der Zeile oder ein
!  Ausrufezeichen direkt hinter dem Programmcode (s.u.)

PROGRAM Plotting_sin  ! Programm Name (for convenience)

IMPLICIT NONE         ! alle Variablen /muessen/ deklariert werden

INTEGER :: nsteps
DOUBLE PRECISION :: hw, PI
DOUBLE PRECISION :: maxx, x, deltax, f


WRITE(*,*)'  Plotten einer Sinuskurve f=sin(x)'
WRITE(*,*)'--------------------------------------'
WRITE(*,*)
WRITE(*,*)' Anzahl der Halbwellen als Vielfaches von PI: '
READ(*,*) hw
WRITE(*,*)' Anzahl der Stuetzstellen ein: '
```

```
READ(*,*) nsteps

PI=3.14159d0          !Exponentialschreibweise fuer Fliesskommazahlen
maxx = hw*PI

WRITE(*,*)' Der x-Bereich liegt jetzt zwischen 0 und ',maxx
WRITE(*,*)' Diskretisierung durch ',nsteps,' Stuetzstellen'
WRITE(*,*)
WRITE(*,*)'       x                              f(x)'

deltax = maxx/nsteps
x      = 0.0d0
DO i=1,nsteps
   f = SIN(x)
   WRITE(*,*)x,'  ',f
   x = x + deltax
ENDDO

END
```

Dateiname: sinus_plot.f90

Die Bedeutung von `implicit none`:
Historisch bedingt enthält Fortran 90 (und höher) noch die Möglichkeit zur impliziten Datentypvereinbarung der Vorgängerversionen (Fortran 66/77). Bei der impliziten Datentypvereinbarung wird vom Compiler anhand des ersten Buchstabens eines Variablen- oder Konstantennamens angenommen, dass der zugehörige Datentyp `real` oder `integer` sei. Und zwar gilt bei der impliziten Datentypvereinbarung die Regel: Beginnt der Name mit i, j, k, l, m oder n (oder dementsprechend mit I, J, K, L, M oder N), so wird bei der impliziten Datentypvereinbarung angenommen, dass es sich um eine Variable vom Datentyp `integer` handelt. Falls ein anderer Anfangsbuchstabe vorliegt, so wird vom Compiler angenommen, dass es sich um eine Variable des Datentyps `real` handelt.
***8tung – Gesundheitswarnung:** Dringende Empfehlung zur Vermeidung von völlig unnötigen Programmierfehlern: Variablen stets explizit definieren!! (d.h. jede Variable im Deklarationsteil zusammen mit dem dazugehörigen Datentyp explizit auflisten) und die implizite Typvereinbarungregel von vornherein ausschalten durch `implicit none` unmittelbar hinter der (z.Bsp.) program-Anweisung.*

① Schreiben Sie das Programm mit emacs und kompilieren es.

② Leistet das Programm das Geforderte? (sinnvolle Testfälle ausdenken!)

③ Ändern Sie das Programm so ab, dass die Ausgabe sinnvoller ist (siehe ②) (Kriterium für sinnvoll: kann ich die Daten in dieser Form plotten?)

④ Üben von if-Abfragen: Ergänzen Sie das Programm um eine Abfrage, die prüft, ob `hw` grösser 0 ist. Befehl für bedingte Ausführung:

```
IF(a.gt.b) THEN
  ! code fuer a>=b
ENDIF
```

oder

```
IF(a.gt.b) THEN
  ! code fuer a>=b
ELSE
  ! code fuer a<b
ENDIF
```

Als Operatoren für Vergleiche können verwendet werden: `.le.`, `.lt.`, `.ge.`, `.gt.` und `.eq.`. `l`'s stehen für "less", `g`'s stehen für und `e`'s für "equal".
In Fortran 90 kann man auch $==$, $/=$, $>=$, $<$ usw. verwenden.

⑤ Die Werte, nach denen das Programm fragt, können auch aus einer Datei umgeleitet werden $<$ - Operator. Erzeugen Sie ein Datei mit Ihren Antworten: Name z.Bsp. `start.dat`:

```
5
100
```

8tung jeweils [ENTER] am Ende der Zeilen eingeben.
Einlesen aus der Datei erfolgt dann mit: `./plot_me<start.dat`

⑥ Ausgabe des Programms vom Bildschirm in eine Datei umleiten. Beachten Sie, dass auch die Fragen des Programms nach der Halbwellenanzahl etc. ebenfalls in der Datei landen. Sie müssen dann "blind" antworten oder mit `<start.dat` die Antworten aus der Datei `start.dat` umleiten .
Nur Ausgabe umgeleitet: `plot_me > plotdaten.dat` erzeugt eine Textdatei mit der Programmausgabe. (Sie müssen "blind" antworten).
Aus-und Eingabe umgeleitet: `./plot_me<start.dat>ausgabe1.dat`

Schauen Sie sich diese Datei mit emacs oder `more` an. (Kommando `more` siehe VL-Folien oder `man more`.

⑦ Speichern Sie das Programm unter neuem Namen ab. Wir möchten nun die Ausgabe *direkt* in eine Datei schreiben.
Öffnen der Datei:

```
 OPEN(UNIT=12,FILE='kurve.dat',STATUS='UNKNOWN',&
&ACCESS='SEQUENTIAL',FORM='FORMATTED')
```

Schreiben in die Datei:

```
 WRITE(12,*) x,'  ',f
```

Schließen der Datei:

```
 CLOSE(12)
```

Schauen Sie sich z.Bsp. mit `emacs` den Inhalt der Datei an und plotten Sie diese mit `gnuplot`.

— ENDE VON ÜBUNG 1 —