Computational Chemistry
- Einführung in Linux und Fortran 77 -

Thorsten Tönsing

E-Mail: thorsten.toensing@uni-bielefeld.de
(Raum: F4-121)

Vorlesungsmaterial

https://www.uni-bielefeld.de/fakultaeten/chemie/ag/tc/lehrveranstaltungen/cc1/

Universität Bielefeld, TC

V.2022

Übersicht (Einführung)

- Einführung in Linux / UNIX
  - System / Login
  - Dateisystem, Verzeichnisse
  - Befehle der shell (CLI)
  - grafische Programme (GUI)
  - Editoren
- Einführung in Fortran 77
  - Programmstruktur
  - Sprachbeschreibung
  - Beispiele
- Übungsaufgaben

System / Login

- Multitasking
- Multi-User System
- Netzwerkarchitektur
  - Zentrale Benutzerverwaltung (IDM)
  - Zentrale Dateiverwaltung → globaler Home-Bereich
    vgl. Windows: Netzlaufwerk → BITS: Laufwerk P:

- Username: ccprakt* *=1..25
- Password: ***
- Passwort ändern: Shell / Terminal -> passwd
- Passwort-Regeln des IDM (IDentity Management):
  - Min. Länge: 10 (länger ist besser)
  - Min. Zeichenklassen: 2 (a-z,A-Z,0-9,Sonderzeichen)
  - Fehleingaben: max. 4, Resetintervall 1 Min., Sperre 45 Min.

Remoteverbindung mit TC-System

- Zugang nur über Secure Shell Protokoll (ssh)
  - Zugangsrechner: goliath.chemie.uni-bielefeld.de
  - goliath ist nur für den Zugang.
    goliath ist NICHT zum Rechnen.
- Login per Username und Passwort
  - IP goliath unbekannt → Captcha.
  - Es gelten die Passwort-Regeln des IDM.
  - Zusätzlich gilt für ssh:
    - Max. 5 Fehlversuche bei richtigem Usernamen
    - Max. 3 Fehlversuche bei falschem Usernamen
    - IP wird dann für 24h gesperrt
    - bei Wiederholung wird die IP dauerhaft gesperrt.
- Kurze Rechnungen auf tc1. (per ssh tc1)
- Lange Rechnungen per Skript auf tcP2A0[a-d]

Secure Shell (ssh)

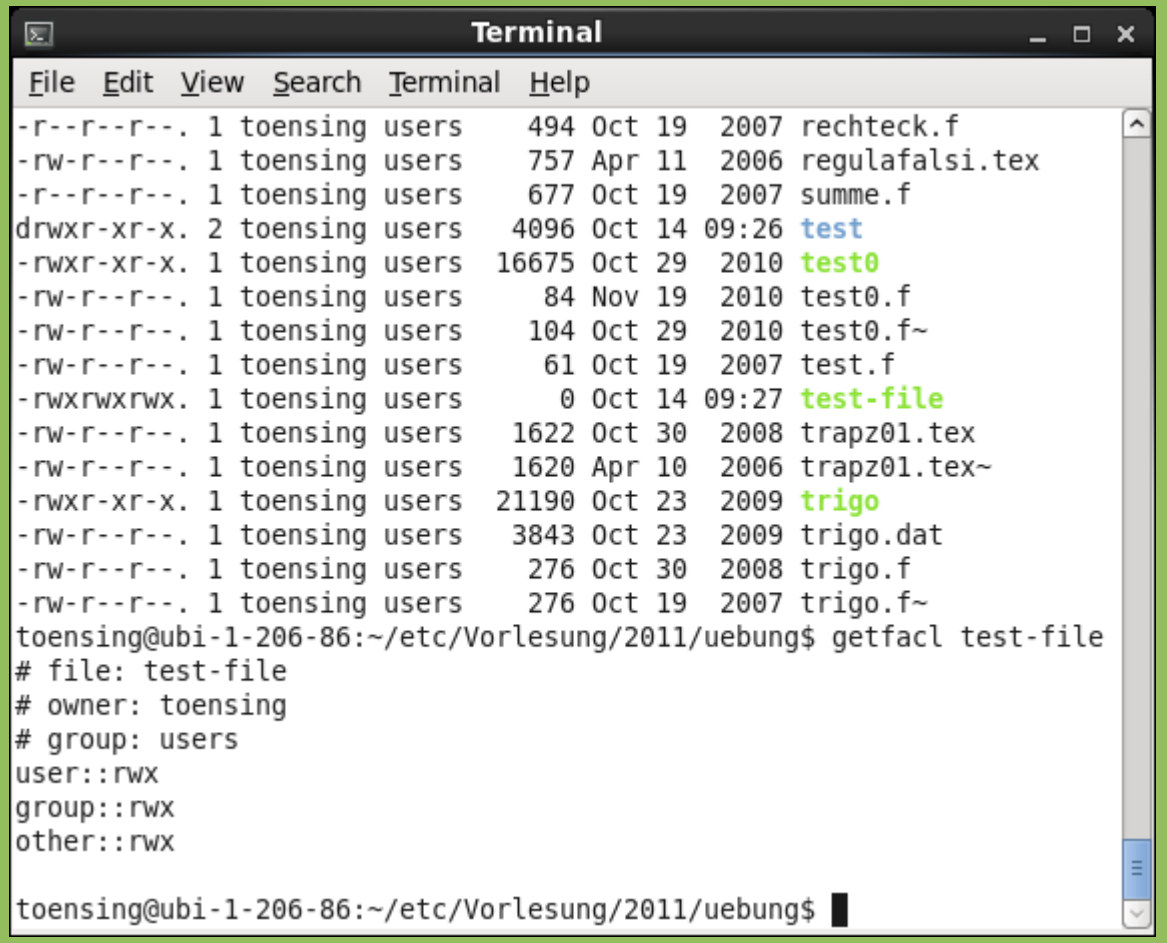
- Linux: Terminal / Shell öffnen.
  Windows 10 ab Version 1809: Eingabeaufforderung (cmd) oder Powershell öffnen.
- Um als ccprakt1 eine Shell auf goliath zu öffnen folgenden Befehl verwenden:
  ssh ccprakt1@goliath.chemie.uni-bielefeld.de
  - Linux: Mit dem Parameter **-X** können auch grafische Programme ausgeführt werden.
  - Windows: Windows benötigt für grafische Linux-Programme einen externen X-Server,
    z. B.: VcXsrv (→ Softwareliste.pdf)

Netzverzeichnis/-laufwerk

- Linux: Das Homeverzeichnis und die Unterverzeichnisse lassen sich mit **sshfs** als Netzverzeichis in die Baumstruktur einhängen und mit **fusermount -u** wieder aushängen.
- Windows (nicht empfohlen, da nicht stabil): Um Netzlaufwerke per ssh verwenden zu können müssen zwei Pakete installiert werden:
  SSHFS-Win https://github.com/billziss-gh/sshfs-win
  WinFsp https://github.com/billziss-gh/winfsp
  Netzlaufwerk Ordner:
  \\sshfs\ccprakt1@goliath.chemie.uni-bielefeld.de
- Mit dem Befehl **scp** lassen sich Dateien zwischen goliath und dem lokalen Rechner kopieren.
  Siehe man scp. Für Windows gibt es zusätzlich WinSCP als grafische Variante.

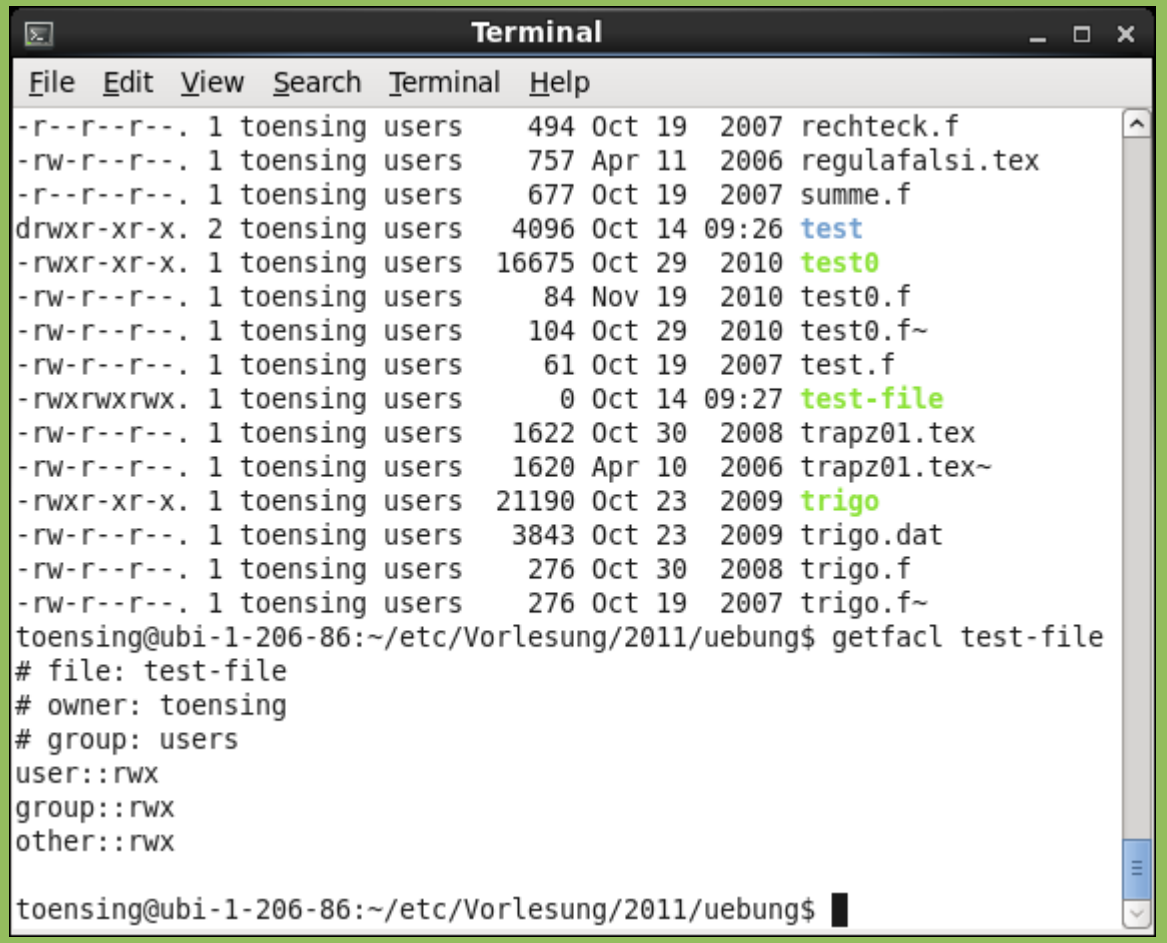
Der List Befehl

- ls, ll (, dir)
  - Permissions
  - # Links
  - Eigentümer
  - Gruppe
  - Dateigröße
  - Änderungsdatum
  - Dateiname
  - ll = ls -l
- Wildcards
  - *
  - ?
  - [Kk],[A-Z],[!A-Z]



Dateien & Verzeichnisse

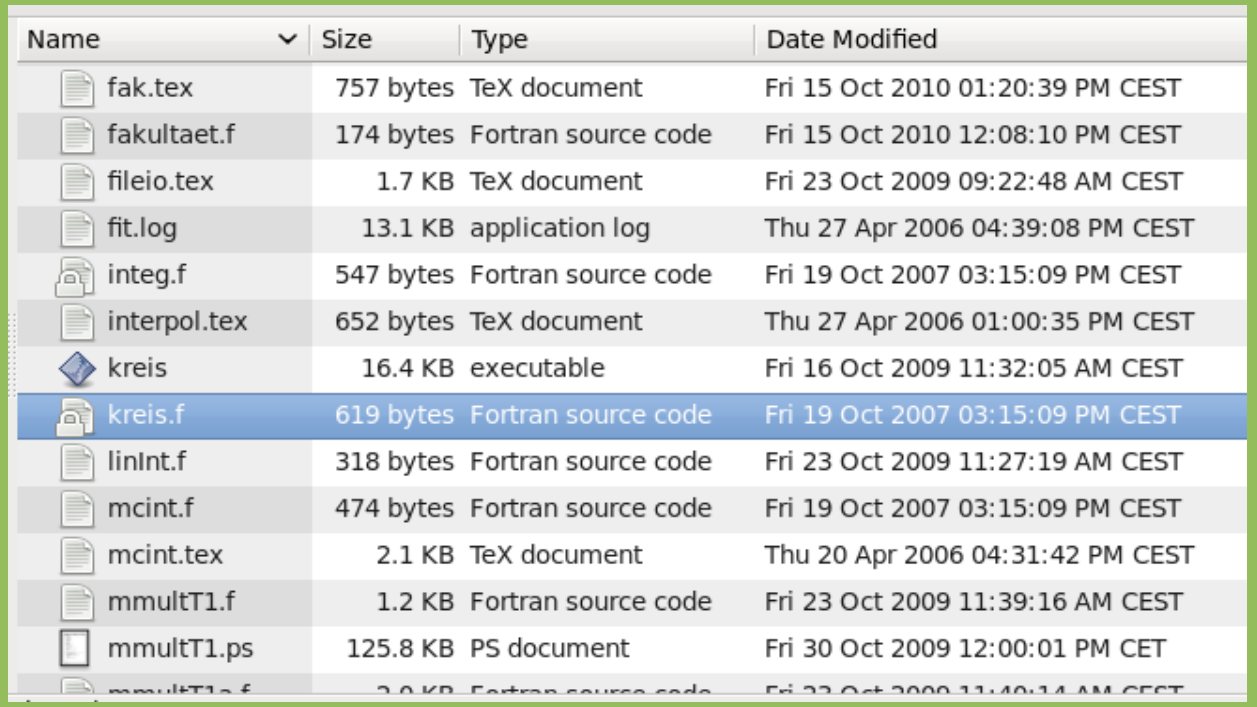
- Zugriffsrechte / Permissions (tuuugggooo.)
  - t – type: - file, l link, d directory, …
  - u - User, g - Group, o – Other:
    - r – read
    - w – write
    - x – execute
    - (t – sticky)
    - (s – set ID)
  - (. - No ACL)
  - (+ – ACL)
    ACL =
    Access Control Lists



Dateien & Verzeichnisse

- Dateitypen
  - Programme / Ausführbare Dateien (Executables)
  - Shell-Skripte
  - Quellcode (Fortran, C, Pascal)
  - Inputs, Outputs
  - Postscript
  - Daten

$ file Datei



Name	Size	Type	Date Modified
fak.tex	757 bytes	TeX document	Fri 15 Oct 2010 01:20:39 PM CEST
fakultaet.f	174 bytes	Fortran source code	Fri 15 Oct 2010 12:08:10 PM CEST
fileio.tex	1.7 KB	TeX document	Fri 23 Oct 2009 09:22:48 AM CEST
fit.log	13.1 KB	application log	Thu 27 Apr 2006 04:39:08 PM CEST
integ.f	547 bytes	Fortran source code	Fri 19 Oct 2007 03:15:09 PM CEST
interpol.tex	652 bytes	TeX document	Thu 27 Apr 2006 01:00:35 PM CEST
kreis	16.4 KB	executable	Fri 16 Oct 2009 11:32:05 AM CEST
kreis.f	619 bytes	Fortran source code	Fri 19 Oct 2007 03:15:09 PM CEST
linInt.f	318 bytes	Fortran source code	Fri 23 Oct 2009 11:27:19 AM CEST
mcint.f	474 bytes	Fortran source code	Fri 19 Oct 2007 03:15:09 PM CEST
mcint.tex	2.1 KB	TeX document	Thu 20 Apr 2006 04:31:42 PM CEST
mmultT1.f	1.2 KB	Fortran source code	Fri 23 Oct 2009 11:39:16 AM CEST
mmultT1.ps	125.8 KB	PS document	Fri 30 Oct 2009 12:00:01 PM CET

Dateien & Verzeichnisse

- Verzeichnisse
  - Baumstruktur → Hierarchie
  - Wurzelverzeichnis (/)
  - Heimatverzeichnis (Home-Directory,~) z.B. für ccprakt1:
    ~ = /raid/home/ccprakt1
  - /raid/home/exchange/praktikum
  - . = aktuelles Verzeichnis (pwd)
  - .. = übergeordnetes Verzeichnis
- Relativer-/Absoluterpfad
- Groß-/Kleinschreibung wichtig!
- Windows: Laufwerk+Baumstruktur, Groß-/Kleinschreibung egal

Befehle der shell

- für Verzeichnisse
  - ls Verzeichnis
  - ll Verzeichnis
  - cd Verzeichnis
  - mkdir Verzeichnis
  - rmdir Verzeichnis
- für Dateien und Verzeichnisse
  - cp Quelle Ziel
  - mv Quelle Ziel
  - rm Datei
  - chmod optionen Datei/Verzeichnis

Befehle der shell

- zum anzeigen / auswerten von Dateien
  - (cat Datei)
  - (more Datei)
  - less Datei
    - Suchen mit „/“, weiter suchen mit „n“, ende mit „q“
  - head Datei
  - tail Datei
  - grep optionen muster Datei
  - file Datei
  - diff Datei1 Datei2
- Hilfe / Manuals
  - man Befehl
  - man *-k* Befehl
    - Verwendet zu Anzeige den less Befehl

Befehle der shell

- Drucken
  - lpstat -a
  - lp -d druckerschlange datei
  - lpr -P druckerschlange datei
- Prozessverwaltung
  - &
  - Ctrl-Z bzw. Strg-Z
  - bg
  - fg
  - ps
  - kill Prozessnummer
  - top

Befehle der shell

- IO-Umlenkung (Standard Ein-/Ausgabe)
  - <,<< (Datei als Eingabe / Input für Programm)
    $ cat < file.txt
  - >,>> (Ausgabe / Output in Datei umleiten)
    $ cat file1.txt > file2.txt
    $ cat file1.txt >> file2.txt
  - | (Pipe: Ausgabe eines Programms als Eingabe für ein anderes Programm)
    $ ll | sort -n -k 5 | tail -5
  - Kombination:
    $ ll | sort -n -k 5|tail -5 > dir5.txt
  - Abzweigung (T-Stück):
    $ ll | tee file1.txt | tail -5

Programme in der GUI

- Linux:ssh mit Option -X / Windows: VcXsrv
- Datei-Browser / File-Manager: nautilus, dolphin
- Shell, Terminal:
  - gnome-terminal
  - xterm
- Editoren
  - gedit, geany(IDE)
  - emacs (auch ohne Grafik im Text-Terminal (CLI))
  - vi (ohne Grafik im CLI)
- Document Viewer: evince (ps,pdf,dvi,...)

Editoren

- vi
  - Modi
    - Eingabemodus
    - Befehlsmodus (<ESC>)
  - i – insert
  - a – append
  - x – delete
  - dw – delete word
  - dd – delete line
  - . – repeat
  - yy – yank line
  - p – paste
  - r – replace
  - 5G – go line 5
  - :w – write
  - :q – quit
  - /text – suche „text“ vorwärts
  - n – wiederhole Suche
- Emacs
  - Control: C-x = Strg/Ctrl-x
  - Meta: M-x = Alt-x
  - C-x C-c – Exit
  - C-x C-s – Save
  - M-g g – Goto
  - Referenzkarte: https://www.gnu.org/software/emacs/refcards/index.html
- gedit
  - Ctrl-S - Speichern
  - Ctrl-F - Find
  - Ctrl-I – Goto
  - Edit → Preferences
    - Display line numbers
    - Right Margin → 73
  - Farbgebung / Schlüsselworte
- IDE (Integrated Development Environment)
  - geany
  - code (Visual Studio)

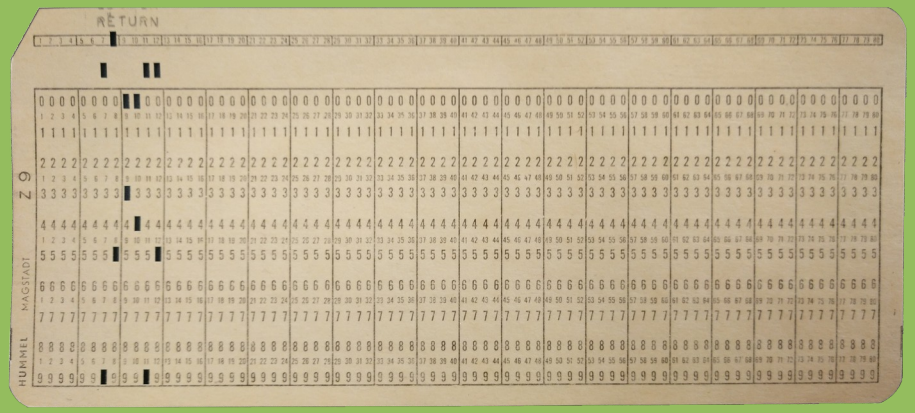
Remote Programmierung

- Schnelle (synchrone) Internetverbindung
  - Linux: ssh -X goliath; ssh -X tc1
  - Windows: VcXsrv → goliath; ssh -X tc1
- Langsame (asynchrone) DSL-Verbindung
  - Bandbreite sparen, X-Server nicht verwenden
  - 1. Konsolen Editor (vi / emacs): ssh goliath; ssh tc1
  - 2. lokaler grafischer Editor
    - Benötigte Ressourcen:
      - Shell auf tc1: ssh goliath; ssh tc1 (ohne -X = kein X-Server)
      - Netzlaufwerk/Netzverzeichnis (sshfs)
      - Lokaler Editor
    - Quellcode erstellen/bearbeiten:
      - Mit lokalem Editor über Netzlaufwerk Quellcode-Datei auf goliath bearbeiten
    - Quellcode compilieren:
      - Mit shell von tc1: Quellcode-Datei auf tc1 übersetzen
    - Programm ausführen
      - Compiliertes Programm in shell von tc1 ausführen

Fortran 77 - Programmstruktur

- Spalteneinteilung (in f95 auch Freiformat)

Spalte	Inhalt	Bedeutung
1	C oder * (oder !)	Kommentarzeile
1-5	Zahl (1-99999)	Anweisungsnummer/Label
6	Leer	
6	Zeichen	Fortsetzungszeile
7-72	Anweisung	Anweisung
73-80	Zeichen	Kommentar



- Genereller Aufbau
  - Kopf: Program Name
  - Deklaration
  - Ausführungsteil (EVA)
  - Subroutinen / Funktionen

```
      program kreis
c     DEKLARATION
c     AUSFUEHRUNGSTEIL
c     Eingabe
c     Verarbeitung
c     Ausgabe
      end
c     SUBROUTINEN / FUNKTIONEN
```

- https://de.wikibooks.org/wiki/Fortran:_FORTRAN_77

Deklarationen

- Datentypen
  - implicit none
  - logical l
  - character c*80
  - integer i (-32768..32767)
  - Integer*8 j (-2147483648..2147483647)
  - real r
  - real*8 dp
  - complex cx,cx2
- data a,b,c /3,1.,5./
- Konstante
  - integer n,jmax
  - parameter (n=10, jmax=22)
- Felder v1:1..20, v2:0..20 und m:1..5,1..10
  - real v1(20), v2(0:20), m(5,10)

```
      program kreis
c     DEKLARATION
      implicit none
c     Konstanten
      real pi
      parameter (pi = 3.14159)
c     Variablen
      real r,flaeche
c     AUSFUEHRUNGSTEIL
c     Eingabe
c     Verarbeitung
c     Ausgabe
      end
c     SUBROUTINEN / FUNKTIONEN
```

Ausführungsteil

- Ein-/Ausgabe
  - print '(format)'. daten
  - print*, daten
  - read*, daten
- Zuweisungen (=)

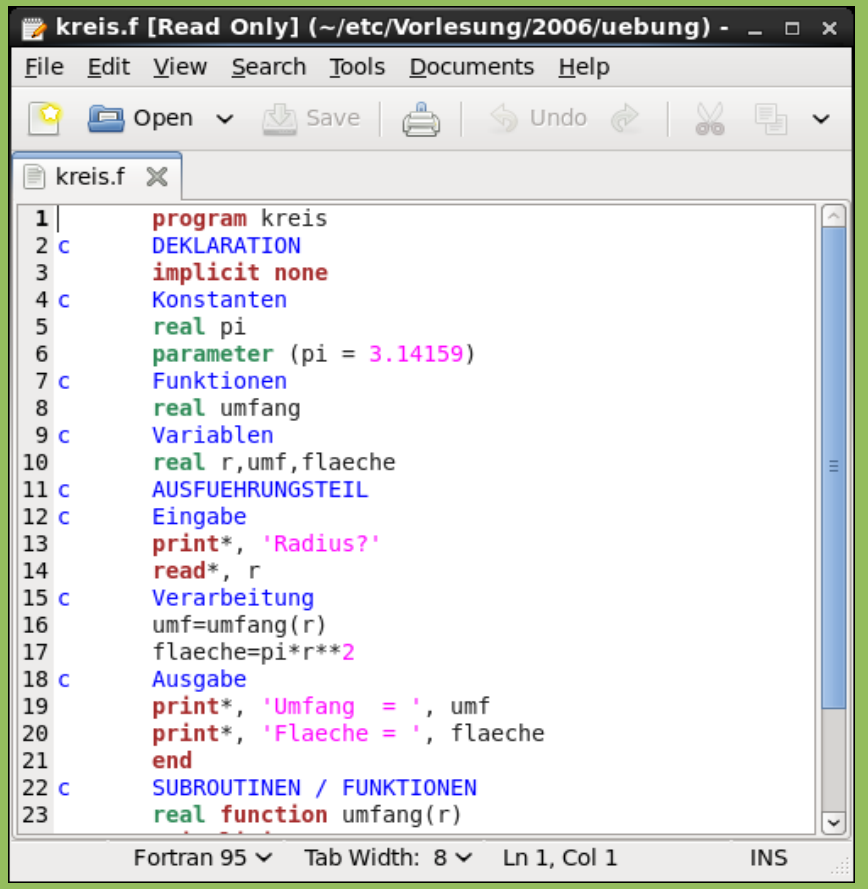
Datentyp	Beispiel
logical	l=.true.; l=.false.
character	c='Hallo'
integer	i=+123; i=0; i=-4732
real / real*8	r=1; r=1.; r=.4; r=-1.4; r=1.4e3
complex	cx=(1.3,7.4)
Felder	v1(3)=7; m(2,2)=3.1

- Implizite Funktionen
  - abs, exp, log, log10, sin, cos, tan, asin, acos, atan, sqrt
  - +, -, *, /, ** (x**y = x^y)
- Eigene Funktionen (Unterprogramm mit Rückgabe)
  - real FUNCTION f(a,b,c)
    real a,b,c
    f=a
    END

Beispiel: kreis.f

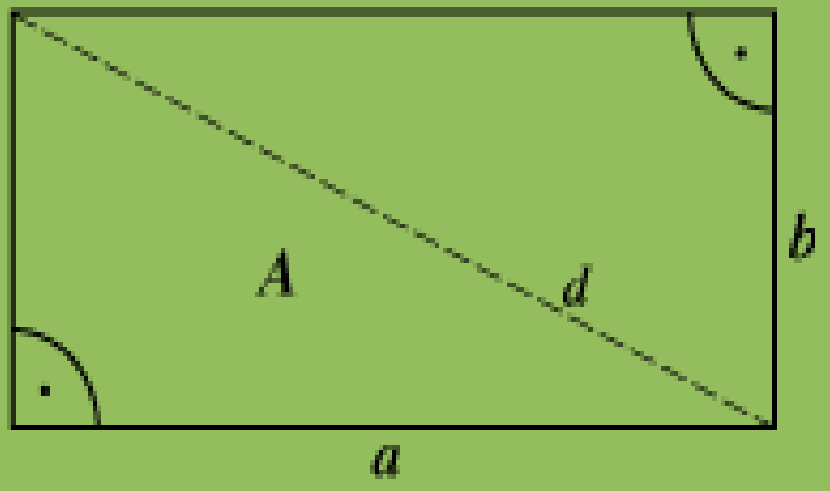
```
      program kreis
c     DEKLARATION
      implicit none
c     Konstanten
      real pi
      parameter (pi = 3.14159)
c     Variablen
      real r,flaeche
c     AUSFUEHRUNGSTEIL
c     Eingabe
      print*, 'Radius?'
      read*, r
c     Verarbeitung
      flaeche=pi*r**2
c     Ausgabe (+ Verarbeitung)
      print*, 'Umfang  = ', 2.*pi*r
      print*, 'Flaeche = ', flaeche
      end
```

- Compileraufruf:
  gfortran -o kreis kreis.f
  f95 -o kreis kreis.f
- Quellcode:
  /raid/home/exchange/praktikum/kreis.f



Aufgabe

- Schreiben Sie ein Programm, das die Länge und Breite einliest und die Fläche, den Umfang und die Länge der Diagonalen eines Rechtecks ausgibt
  - Testen Sie das Programm am Beispiel: Länge = 4, Breite = 3



Schleifen / Verzweigungen

- DO-Schleife Bsp: summe.f

```
do i=0,10,2
  print*, i
enddo
```

- GOTO

```
99   code
     code
     goto 99
```

- Vergleichsoperatoren
  - .lt. → <, .le. → <=
  - .eq. → =, .ne. → ≠
  - .gt. → >, .ge. → >=
- logische Operatoren
  - .not., .and., .or.
  - .eqv. (equivalent = gleich)
  - .neqv. (not equivalent)

- if-then Bsp: if-block.f
  - if (a.lt.b) then
    print*, “a kleiner b“
    endif
  - if (a.lt.b) then
    print*, “a kleiner b“
    else if (a.gt.b) then
    print*, “a groesser b“
    else
    print*, “a gleich b!?“
    endif
- if
  - if (a.lt.b) print*, “a kleiner b“

Wahrheitstabelle

.not.	w	f
	f	w

.and.	w	f
w	w	f
f	f	f

.or.	w	f
w	w	w
f	w	f

.eqv.	w	f
w	w	f
f	f	w

.neqv.	w	f
w	f	w
f	w	f

Eigene Funktionen

- Deklaration
  ```
  real f
  ```
- Implementation
  ```
  real FUNCTION f(a,b,c)
  real a,b,c
  f=a
  END
  ```

Funktionen:

Variablen bilden abgeschlossene Welt mit Austauschmöglichkeit über Parameter und Rückgabewert (und COMMON-Blöcken; später).
Die variable i in mySum ist unabhängig von der Variablen i im Hauptprogramm.

```
 1          program summe
 2          implicit none
 3          integer i,k,mySum
 4          print*,"Untere Grenze?"
 5          read*, i
 6          print*,"Obere Grenze?"
 7          read*,k
 8          print*,"i=",i
 9          print*,"Summe:",mySum(i,k)
10          print*,"i=",i
11          end
12
13          integer function mySum(a,b)
14          implicit none
15          integer a,b,i,s
16          s=0
17          do i=a,b
18           s=s+i
19          enddo
20          mySum=s
21          end
```

$$mySum(a,b)=\sum_{i=a}^{b} i$$
$$=a+(a+1)+...+b$$

Beispiel Schleife: a=1, b=3:

Zeile	16	17	18	17	18	17	18	17	20
a:	1	1	1	1	1	1	1	1	1
b:	3	3	3	3	3	3	3	3	3
i:	?	1	1	2	2	3	3	4	4
s:	0	0	1	1	3	3	6	6	6

Aufgaben

- Schreiben Sie ein Programm, das die ganzen Zahlen von n bis m (mit n <= m) ausgibt, wobei die Grenzen n und m vom Programm eingelesen werden.
  (do-Schleife, keine FUNCTION)
- Erweitern Sie das Programm so, dass auch n > m berücksichtigt wird und die Zahlen bei n > m absteigend ausgegeben werden.
  (if-Bedingung & do-Schleife mit negativem Inkrement)

Subroutine

Unterprogramm ohne Rückgabewert

- (Keine Deklaration)
- Implementation:

```
subroutine sub(a,b)
  real a,b
  Anweisungen
end
```

- Aufruf

```
call sub(c,d)
```

Parameter in Function und Subroutine:

- call by reference (bsp: test-callby-ref.f)

Deklaration statischen Speichers

- DATA a,b,c,d,e,f /1,2,4*7/ Bsp: data-save.f
  - Anfangswerte: a=1,b=2,c..f=7
    real r(10,10)
    data r/50*5.0,50*75.0/
- SAVE a,b,c
  - Unterprogramme: Inhalt von Variablen bleibt erhalten
- Common-Block Bsp: common-bsp.f

  (Datenaustausch über gemeinsamen Datenbereich)

```
common a,b
common /name/ c
real a,b,c
...
subroutine foo(u)
common x,y
common /name/ z
real x,y,z
```

Einfacher Dateizugriff

- OPEN([unit=] unitid
  [, access='sequential' | 'direct']
  [, err= label] [, iostat=ios]
  [, file= filename]
  [, status='unknown' | 'old' | 'new' | 'scratch'])

  open(10,err=45,file='eingabe.dat')
  open(40,err=55,file='ausgabe.dat',status='new')

- CLOSE([unit=] unitid
  [, iostat=ios] [,err=label] [,status='keep' | 'delete'])

  close(10)
  close(40)

Einfacher Dateizugriff

- READ(unitid, *|'(format)' [,iostat=ios] [,err=label] [, end=label]) [iolist]

  read(10,*,end=100,err=201) var1,var2

- WRITE(unitid, *|'(format)' [,iostat=ios] [,err=label]) [iolist]

  write(40,'(F15.7)') var1

- Format:[n]A[w],[n]Dw.d,[n]Ew.d,[n]Fw.d,[n]Iw[.m],[n]Lw
  A-char, D-real*8(exp), E-real(exp), F-real,
  I-integer, L-logical
  n-wiederh. (optional), w-Feldweite,
  d-Nachkommastellen, m-Mindestziffern (optional)

Beispiel: Dateizugriff (Ausgabe)

- Programm:

```
1         program FileOutput
2         implicit none
3         real*8 fMagic
4         fMagic=42
5         open(10,file='fileout.output',status='new',err=100)
6         write(10,*,err=100) 'Hallo Welt.'
7  c2345678
8         write(10,'(1A20,D15.8,E15.8F15.8)',err=100)
9       *  'Die Zahl lautet:', fMagic, fMagic, fMagic
10        close(10,err=100)
11        goto 900
12 100    print*, 'Bei der Ausgabe ist ein Fehler aufgetreten.'
13 900    end
14
```

- Inhalt von fileout.output und Formatanweisung: