Програмиране в UNIX среда

Интерпретатори, компилатори. Изпълними файлове.
Програмиране под UNIX. Система от компилатори GCC.
Граматика на език за програмиране.

Програмни езици

- How computers work.
- Machine code.
- High level languages.
- Fortran : FORMula TRANslation.
- Other languages :
  - Basic
  - C/C++
  - Pascal

Програмни езици

Програмни езици

- **FORTRAN**

FORTRAN 77 Programming.

Програмен цикъл

- Анализ на задачата.
- Планиране на програмата – структурен подход.
- Flowcharts & *Dry running*.
- Редакциа на програмния код.
- Компилиране и свързване (линкване) на програмата.
- Изпълнение и дебъгване (debug) на програмата.
- Редакция и прекомпилиране.

G77 компилатор

- Edit source program (*.f) with “emacs” editor. Save file.
- Compile and run on Unix command line in a *shell* window :
  - |litov@heph> g77 –o test test.f
  - |litov@heph> test

Структура на програма на FORTRAN

- Program name.
- Declare variables and structures.
- Assign values to variables.
- Process data.
- Print results.
- End program.

Flow of a Program.

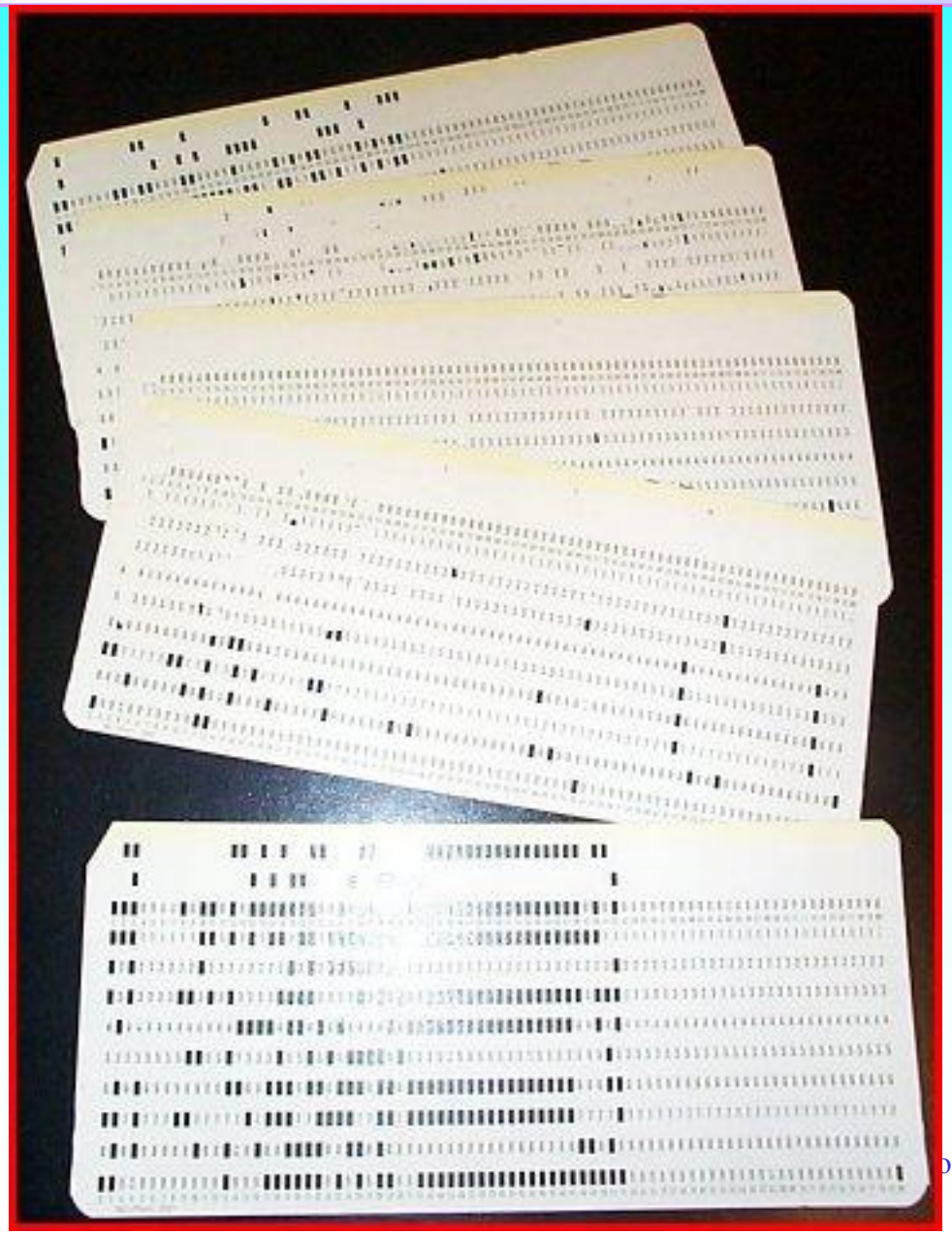
- Linear sequence.
- One command per line.
- Position on line : **Very Important!**
- Comments Statements (ignored).
- Repetition : Loops.
- Selections : Conditional statements.
- Always finish with an **END** statement.

Position on a line.

- The layout of FORTRAN program dates back to old 80 column *punched cards*, which were used for program input.

1	2-5	6	7-72	73-80
			Total=x_value+y_value	
		&	+z_value	
C			Comment line.	
9	9999		FORMAT('Answer =',I4)	

Карти с програмен код



Декларация на променливи

- Variable names :
  - Must be at least one alphabetic character long, up to a maximum of 31 alphanumeric characters.
  - Must start with an alphabetic character. Case insensitive.
  - Alphanumeric characters are : a-z, 0-9 and the underscore (_).
  - Implicit variables. I to N integers!

Примери

- Валидни имена :
  - X
  - THEDAY
  - Min_cur
  - Time28
- Невалидни имена :
  - X*Z
  - THE TIME
  - 7YEARS
  - _no_way$

Основни типове данни

- REAL x=5.0
- INTEGER i=20
- COMPLEX z=(1.4,3.2)
- LOGICAL test=.TRUE.
- CHARACTER char='Hello'

- More advanced data types can be made from these *basic* types.

Декларации

- <Data Type> <variable> [,<variable(s)>]
- e.g.

- REAL x
- REAL radius,volume
- INTEGER loop,temp
- CHARACTER string*10,name*30

Параметри

- Parameters are constants, their value, once defined, can not be changed.

- REAL g,pi
- INTEGER days

- PARAMETER (days=365)
- PARAMETER (g=9.81,pi=3.142)

Присвояване на стойност

- <variable> = <value> | <variable> | <expression>

- radius=2.5
- y=z
- test=value+loop-temp
- volume=(4.0*pi*radius**3.0)/3.0

- Expressions follow the *BODMAS* precedence rule. Operators +, -, *, / and **

Control Structures.

- Basic building blocks of programs.
- They control the flow of the program.

- There are 3 different types :

- Linear Sequence.
- Selection.
- Iteration or Loop.

Other Statements.

- PROGRAM [*program name*]
- END

- C or * A comment.

- PRINT*,'Hello'
- PRINT*,'Value of X = ',x
- This is *free format* output.

Вход на данни

- Programs are useless without data!
- Use the READ statement to allow users to input data.
- Prompt user for the data too!
- e.g.
- PRINT*,'Enter values for x & y :'
- READ*,x,y

Character Input.

- A normal read statement can not be used to enter character variables. Use the following:

- PRINT*,'Continue (y/n) : '
- READ '(A1)',yes_or_no

- '(A<n>)' – <n> is the number of characters.

Добър програмен стил

- Comment your program!
- FORTRAN keywords in upper case.
- Variables in lower case.
- Use descriptive variable names.
- Blanks may be used to improve readability.
- Indent code with “tabs”.

General Program Layout.

- **PROGRAM [** *program name* **]**
- **[** *comments* **]**
- **[** *declaration statements* **]**
- **[** *executable statements* **]**
- **STOP**
- **END**

Логически променливи

- To define use :
- LOGICAL test
- Assignments :
- test = .TRUE.
- test = .FALSE.
- Numerically 0 is False, 1 is true.
- On IRIX : PRINT*,test gives **T** or **F**.

Логически изрази

- A logical expression is made up of :
  - Variables.
  - Constants.
  - Logical operators.

  - e.g.
  - radius .EQ. 24
  - radius .NEQ. old_radius

Логически операции

- Test conditions :
  - .LT. less than.
  - .GT. greater then.
  - .LE. less than or equal to.
  - .GE. greater than or equal to.
  - .EQ. equal to.
  - .NE. not equal to.

Логически операции

- Logical Test Operators :
  - .AND.
  - .NOT.
  - .OR.
  - e.g.
  - (a .LE. 3) .OR. (b .GE. 5)

Условия

- Conditional program flow.
  - IF [*condition(s)*] THEN
  - [*statement(s)*]
  - ELSE IF [*condition(s)*] THEN
  - [*statement(s)*]
  - ELSE
  - [*statement(s)*]
  - END IF

Примери

```
IF ( test  .EQ.  2 )  x=12

IF  ( test  .GT.  5 )  THEN
                       PRINT*,test,' is > 5.'
 ELSE
                       PRINT*,test,' is < 5!'
END IF
```

Примери

```
IF ( day  .EQ.  'Monday' ) THEN
                <statement(s)>
ELSE IF ( test  .EQ.  .TRUE. ) THEN
                <statement(s)>
ELSE IF ( ( x .GT. 4 ) .AND. ( y .LT. 3 ) ) THEN
                <statement(s)>
ELSE
                <statement(s)>
END IF
```

Вградени (intrinsic) функции

- Functions return a value of certain data type.
- Parameters are passed to functions.
- Parameters can be variables, constants or expressions.
- Intrinsic means “built-in”.

  - y=ATAN(1.0) y=SIN((pi*degrees)/180.0)
  - y=LOG10(x) y=x+3.0*LOG(x)
  - y=EXP(2.4) y=INT(2.3)

- More examples…

SIN(x)	ASIN(x)
COS(x)	ACOS(x)
TAN(x)	ATAN2(x)
SINH(x)	EXP(x)
COSH(x)	LOG(x)
TANH(x)	LOG10(X)
SQRT(x)	
ABS(x)	
MAX(x1,x2…)	Max value of x1, x2, …

Масиви (Array Variables)

- An array is a simple structure, capable of storing many variables of the same type in a single data structure.
- Declare arrays with other variables.
- Arrays can be of any data type.
- Declaration :

- REAL array(100)
- INTEGER loop(20),data(500)
- CHARACTER names(10)*20

Масиви

- Each array member is addressable by an array *subscript.* (A, A^{i}, $A^{i,j}$, $A^{i,j,k}$)
- e.g for REAL x(10), s*ubscripts* range from 1 to 10. i.e. array(1), … array(10)
- The subscript may be a constant, variable or expression :

- array(1)=SIN(x)
- array(loop)=4.0*array(loop-1)

Масиви

- Consider :
- REAL array(10)
- As mentioned before, array subscripts range from 1 to 10.
- We can over ride this default:

- REAL array(-10:10)
- INTEGER data(0:200)

Масиви

- Arrays can be multi-dimensional
- **REAL array(5,2)**
- i.e. array(1,1), array(1,2), array(2,1)…
- Taking things to an extreme!!!
  - **REAL array(5,0:12,20,-10:100)**

Iterations.

- Loops allow blocks of command to be repeated.
- Simplest is the **DO** loop.

- **DO** <index> = <start>,<stop> [,<step>]
- <statement(s)>
- **END DO**

Пример на цикъл DO

```
INTEGER loop
REAL array(100)

DO loop=1,100,1
        array(loop)=0.0
END DO
```

DO WHILE цикъл

- Another type of loop :

- **DO WHILE** <condition>
- <statement(s)>
- **END DO**

- Not in all versions of FORTRAN 77.

Пример

- **INTEGER test**
- **test=15**
- **DO WHILE (test .GT. 12)**
- **test=test-1**
- **END DO**

- Useful for repeating a program?

The Horrible Past…

- **DO** loop with numeric label.

- **DO 1000 loop=0,10,2**
- **<statement(s)>**
- **1000 CONTINUE**

- Implied **DO** loop!

- **READ*,(value(loop),loop=1,10,1)**

The DATA statement.

- Another method of assigning values to variables and arrays.

- **REAL a,b,c**
- **INTEGER d(5),lots(100)**
- **CHARACTER single(5)*1**
- **DATA a /1.5/**
- **DATA b,c /2.5,5.6/**
- **DATA d /1,2,3,4,5/**
- **DATA lots /100*0.0/**
- **DATA single /5*'Y'/**

Advanced Data Types.

- Structures. Not in all versions of F77.
- Structures can hold many variables of different types.
- Structures are declared with all other variables at the beginning of a program.
- With structures you can build powerful, custom data types.

Declaring A Structure.

- **STRUCTURE / book_type /**
- **CHARACTER title*80**
- **CHARACTER author*40**
- **INTEGER stock_level**
- **REAL price**
- **END STRUCTURE**

The Next Step…

- Now we must declare a **record** of the type defined in the structure.
- **RECORD / book_type / science**
- **RECORD / book_type / rubbish(10)**

Value Assignments.

- Each part of the record is identified thus :

- science.title = ‘2001’
- science.author = ‘A.C.Clarke’
- science.stock = 25
- science.price = 9.99

For the More Brave…

- Remember?

- **RECORD / book_type / rubbish(10)**

- **DO loop=1,10**
- **READ'(A80)',rubbish.title(loop)**
- **READ*,rubbish.price(loop)**
- **END DO**

The Story So Far…

- FORTRAN 77 program structure.

- **PROGRAM** <program name>
- <comment(s)>
- <declaration(s)>
- <assignment(s)>
- <statement(s)>
- **END**

Intrinsic or Extrinsic?

- So far we have seen :
- **value = SIN(angle_in_radians)**
- Every angle has to be converted to radians with the same formula. Very repetitive!

This Would Be Nice…

- FORTRAN 77 does not have a degrees to radians function, but one would be useful. e.g.

- **radians = RAD(degrees)**
- or
- **value = SIN (RAD (degrees))**

-

We Can Do It!

- FORTRAN allows you to define your own functions!
- They conform to all the rules of intrinsic functions.
- Must return a single value.
- Defined after the **END** statement of the main program.

A Simple Function.

```
Ø REAL FUNCTION radians(degrees)
Ø       REAL degrees,pi,temp

Ø       pi=4.0*ATAN(1.0)
Ø       temp=(pi*degrees)/180.0
Ø
Ø       radians=temp
Ø RETURN
Ø END
```

More Complex.

```
REAL FUNCTION power(x,y)
        REAL temp,x
        INTEGER loop,y
        temp=1
        DO loop=1,y
                temp=temp*x
        END DO
        power=temp
RETURN
END
```

Subroutines.

- Another method for “structuring” your program.
- Subroutines do not return a value.
- Subroutines may change none, one or many of the parameters passed to them.

Definition.

```
      SUBROUTINE print_at(message,line)
            CHARACTER message*80
            INTEGER loop,line

            DO loop=1,line-1
                  print*
            END DO

            PRINT*,message

      RETURN
      END
```

Calling Subroutines.

- The **CALL** statement is used to call subroutines.
- Parameter data types must match in the main program and subroutine.

- **CALL print_at(‘Hello!’,10)**

Memory Allocation!

- The compiler reserves memory for arrays. Consider…

- CALL bad(100)

- SUBROUTINE bad(n)
- INTEGER n
- REAL array(n)

Локални и глобални променливи

- INTEGER, REAL, CHARACTER, COMPLEX, LOGICAL define local variables, valid only inside PROGRAM or SUBROUTINE or FUNCTION
- Area – DIMENSION B(10), C(5,5),D(1,2,3)
- **Global definition**
- COMMON/name/ A, B(10), C(5,5),D(1,2,3)
- Just put inside PROGRAM or SUBROUTINE or FUNCTION where we are going to use some of these variables
- In this way the values of the variables are transferred from one subroutine (function) to another.
- **Unlabeled COMMON**
- COMMON/ / HBOOK(HLIM)
- Dynamical extension of the memory

Screen Output.

- Ø So far the output of results to the screen
- Ø has been “messy”, because we have been
- Ø using “free format”. e.g.

- Ø PRINT*,’Radius = ‘,radius,’ cm’

- § typical screen output :

- § Radius = 7.2345121 cm

Free Format.

- Free format is simple and easy to use. e.g. with the PRINT* and READ* statements.
- Assumes numeric input, therefore limited.
- Always uses greatest accuracy possible.
- Pads out printed variable and text into columns.
- Lines that should fit on the screen “wrap around” onto the next line, because of this padding.

Formatted I/O.

- The solution to our problem is:
  - Formatted I/O (Input, Output)
- You have seen formatted input
- already!
- READ'(A10)',character_string

Formatted I/O.

- The general form of formatted I/O
- Statements is as follows:

  - PRINT'(<Format>)',<variable(s)>
  - READ'(<Format>)',<variable(s)>
  - <Format> is a format specifier.

Format Specifiers.

- Format specifiers for variables
- consist of a letter and a digit(s).

- A : Character variable.
- I : Integer variable.
- F : Real variable.
- E : Real variable, exponential form.

Examples.

- A10 : String variable 10 characters long.
- e.g. ‘Hello ‘.

- I8 : Integer, 8 digits long.

- F6.2 : Real variable, 2 decimal places, 6 digits
- long including decimal points and minus
- signs.

F6.2 again.

- All these numbers are in F6.2 format.

Number	C1	C2	C3	C4	C5	C6
345.19	3	4	5	.	1	9
1.2			1	.	2	0
-23.45	-	2	3	.	4	5
5			5	.	0	0
9999.99	*	*	*	*	*	*

Примери

- READ’(A30)’,string1
- READ’(A30,2I4),string2,num1,num2

- PRINT’(“Answer = “,F6.2)’,answer

§ “ / ” and “ X “ are new line and space.

- PRINT’(/“A = “,I2,2X,”B = “,F10.1,//)’,a,b

- PRINT’(“Enter a number “,$)’
- READ*,number

The old way…

- Formatted I/O the old way, with
- numeric labels.

```
100  FORMAT(/'A = ',I2,2X,'B = ',F10.1//)
            PRINT 100,a,b

200  FORMAT(3I5)
            READ 200,int1,int2,int3
```

Data Files.

- You must first open a data file.
- Then read or write data.
- Finally close the data file.

- Data files are analogous to books.

- Fortran OPEN statement.

- OPEN(UNIT=x,FILE=y,STATUS=z)

Отваряне на файл с данни

- Choose unit numbers >6
- Unit 5 = keyboard and Unit 6 = screen!

- OPEN(UNIT=20,FILE='data.dat',STATUS='NEW')

- FILE='data.dat'
- FILE='/home/litov/data/data.dat'
- FILE=file_name

- STATUS='NEW'
- STATUS='OLD'
- STATUS='UNKNOWN'

Четене и писане

- Once a data file is opened use READ and WRITE statements to access the data file.
- READ(<unit>,<format>),<variable(s)>
- WRITE(<unit>,<format>),variable(s)>

Пример на файлов вход/изход (File I/O)

- e.g.
- READ(1,*) num1,num2,num3
- WRITE(20,'(5X,I5,10X,3F5.1)') a,b,c,d
- READ(25,'(2F10.5)') data1(loop),data2(loop)

Затваряне на файл с данни

- CLOSE(UNIT=<unit> | <unit>)
- CLOSE(UNT=20)
- CLOSE(20)
- Close data files when you have finished with them!

File Pointer.

- The file pointer is positioned at the beginning of a data file when it is first opened.
- REWIND(UNIT=<unit> | <unit>)
- Moves the file pointer to the start.
- REWIND(20) or REWIND(UNIT=20)

Error Trapping I

- IOSTAT : Used to test if a file exists if opened with ‘OLD’ or ‘UNKNOWN’ status. e.g.

- OPEN(IOSTAT=I,UNIT=20,
- & FILE=‘test.dat’,STATUS=‘OLD’)

- IOSTAT returns an INTEGER value.

Error Trapping II.

- Integer IOSTAT values returned are:

  - 0 : File opened without errors.
  - >0 : Error, file not found?
  - <0 : As condition 0, but at end of file (EOF), file empty.

  - What about EOF during reading data?

Error Trapping III.

- Using the END option with a READ
- Statement you can test for EOF.

```
      DO WHILE ( .NOT. 0 )
            READ(25,'(I5)',END=100 )data(i)
      END DO
100   CONTINUE
```

Литература:

- http://www.wylug.org.uk/talks/2003/04/unix.pdf
- http://ce.sharif.edu/courses/ssc/unix/resources/root/Slides/unixhistory.pdf
- http://www.cs.uga.edu/~eileen/1730/Notes/intro-UNIX.ppt
- http://remus.rutgers.edu/cs416/F01
- http://www.cs.virginia.edu/~cs458/
- http://www.bobbooth.staff.shef.ac.uk/hpcs/materials/material.html
- http://www.comm.utoronto.ca/~jorg/teaching/ece461
- http://home.iitk.ac.in/~navi/sidbilinuxcourse/
- http://www.cs.washington.edu/homes/bershad/Mac/ssh/practicalmagic.pdf
- http://www.cs.cf.ac.uk/Dave/C/CE.html
- http://www.le.ac.uk/cc/tutorials/c/ccccintr.html
- http://www.shef.ac.uk/uni/academic/N-Q/phys/teaching/phy225/index.html

http://www.bobbooth.staff.shef.ac.uk/hpcs/materials/material.html