|  |  |  |
| --- | --- | --- |
| **ACTIES OP HET SCHERM** | **VOICE-OVER** | **DUUR** |
| Start in empty terminal window | Let’s write and run our first Fortran application. It’s nothing fancy, it will simply write “hello world” to standard output, which is shown in the terminal. |  |
| 1. open editor 2. write program-end program 3. write print statement 4. save file | 1. Since a Fortran program is just a text file, you can use any editor you like. I’ll be using vim, since that’s my favorite. The name for the Fortran source file is “hello.f90”. 2. A Fortran program starts with the keyword “program” followed by the name of the program. It ends with “end program”, optionally followed by the name of the compilation unit. I always add the name since that makes the code easier to understand. 3. Now for the actual statement that does the work, the print statement. The keyword “print” is followed by formatting information. For simplicity, we will just use the default, indicated by an asterisk. Next we type what is to be printed, our “hello world” string. Strings in Fortran can be delimited either by single or double quotes. I’ll be using single quotes. 4. We’re done, we can save the file and return to the terminal. |  |
| 1. switch to terminal 2. compile file, use -o 3. run application | 1. Fortran is a compiled language, that means that the source code we just wrote has to be translated into machine code that can be executed by the CPU. That is the job of the Fortran compiler. The one we will be using is gfortran, which is open source and freely available. 2. We use the -o option to specify a name for our application, “hello.exe” in this case. 3. We now have an executable file “hello.exe” which we can run. Lo and behold, the text “hello world” is written to the terminal. |  |
| 1. Switch to editor 2. Turn string into parameter, make typo in name 3. Save file | 1. Let’s change our code a little bit. We go back to the editor and introduce a variable so that our output string is named. This is a string, so the corresponding type is “character” and we have to specify a length. That should be at least the length of our string, but it can be larger, so let’s take 20 to be on the safe side. The name of variable is “message” and we assign its value. 2. Since we’re not going to change “message” in our program, we might as well make it a constant, so we add the keyword parameter. 3. Now we can save the file and return to the terminal. |  |
| 1. Switch to terminal 2. Compile file 3. Run application | 1. We compile the source code again and… 2. Run it. The output is of course the same, but wait… it isn’t. |  |
| 1. Switch to editor 2. Insert “implicit none” 3. Save file | 1. It is good practice to add “implicit none”. This will be discussed in more detail later but in brief, it tells the compiler that all variables have to be declared explicitly. |  |
| 1. Switch to terminal 2. Compile file | 1. When we compile, we get an error. Learning to interpret compiler errors is quite important, so lets look at it carefully. It says that the variable “message” with a single “s” is not defined. True, because we named our variable “message” with double “s”. |  |
| 1. Switch to editor 2. Fix typo 3. Save file | 1. This is of course easy to fix. |  |
| 1. Switch to terminal 2. Compile file 3. Run applicatoin | 1. When we now compile the application… 2. And run it, the output is again as expected. |  |
|  | We have just created our very first and very simple Fortran application, time to move on to more exciting features. |  |
| **TOTALE DUUR** | | *Maak je screencast niet langer dan ca. 6 minuten.* |