Why and how to use getkey

Updated September 24, 2017

When running programs on LINUX and similar OS there is no editing or history available on the command level. On Windows there is normally a possibility to recall previous commands and to edit the command line.

In order to have this facility it is necessary for a program to read character by character from the keyboard without waiting for the user to press the RETURN key. Such a facility is provided by the GETKEY function written in C.

Such a routine can also be used for other purposes for example to allow users to interrupt lenghty calculations or listings without leaving the program and loosing results.

The getkey function used here has been written by Urban Jost and has been downloaded from the web http://www.urbanjost.altervista.org/LIBRARY/libCLI/Getkey/getkey.html

Each OS and terminal can have different ways to navigate the cursor on the screen. At the end of this text there is an extract of the bintxt subroutine that is found in the metlib3.F90 file. You can modify the way to control your editing and you may have to modify the code that moves the cursor one step backward on on your screen. The way to edit input I have implemented follows mainly the emacs standard.

The necessary files are in the utility/GETKEY directory.

To install getkey these are the necessary steps:

1. The C routine getkey() must be compiled with C for your OS. Available are BSD, V13, V13B, LINUX. On a mac use BSD, otherwise you have to test which does not give any error messages.

   ```
   >>> On MAC that is: gcc -c -DBSD getkey-original+testprogram.c
   ```

   If you wish you can use the getkey testprogram, you must then compile with the following options:

   ```
   >>> On MAC that is: gcc -DBSD -DTESTPRG getkey-original+testprogram.c
   ```

   This creates a progran a.out. It is useful to check which code that moves the cursor backward (BACKSPACE) on your screen.

   ```
   >>> On MAC that is: ./a.out
   press keys ('q' to quit)
   C:KEY=j 106
   C:KEY=k 107
   ```

```
C:KEY=n 110
C:KEY=27
C:KEY= 2
C:KEY= 127
C:KEY= 3
C:KEY=c 99
C:KEY=q 113
```

2. When you managed to compile (and test) rename the compiled version of getkey (without the test program) to just getkey.o

   ```
   >>> On MAC that is: gcc -c -DBSD getkey-original+testprogram.c
   >>> Om MAC that is: mv getkey-original+testprogram.o getkey.o
   ```

   Getkey use the Fortran/C iso-C-binding through the interface defined in M_getkey+testprogram.F
   You must use a Fortran compiler compliant with the 2003 Fortran standard like GNU Fortran 4.8 (or later). Compiling M_getkey+testprogram.F90 creates a module m_getkey.mod which is necessary for linking the getkey function to the OC software.

   ```
   >>> On MAC that is: gfortran -c M_getkey+testprogram.F90
   ```

   You can again compile with a small test program:

   ```
   >>> On MAC that is: gfortran -DTESTPRG0 M_getkey+testprogram.F90 getkey.o
   ```

   You have now a new a.out you can test.

   ```
   >>> On MAC that is: ./a.out
    begin striking keys to demonstrate interactive raw I/O mode
    q to quit; up to 40 tries allowed
              1  f03:key=b->          98
              1  f03:key=->           2
              1  f03:key=B->          66
              1  f03:key=          27
              1  f03:key=q->         113
   ```

3. In OC a subroutine routine bintxt is used for all input. This has a plain "read" statement to obtain input from keyboard or macro files. In the special metlib3+getkey.F90 provided in this directory the getkey function is used for keyboard input.

   You may change the characters you want to use for line editing and you MUST check (using the test programs above) which code moves the cursor backward one step on the screen. Then look in the metlib3+getkey.F90 file around lines 3030 to 3260. You will find a section like:

```
! CONTROL CHARACTERS FROM KEYBOARD
! DEL delete current character
    integer, parameter :: ctrla=1        ! CTRLA move cursor to first position
    integer, parameter :: backspace2=2   ! CTRLB move cursor one step left
    integer, parameter :: ctrlc=3        ! CTRLC terminate program
    integer, parameter :: ctrld=4        ! CTRLD delete char at cursor
    integer, parameter :: ctrle=5        ! CTRLE move cursor to last position
    integer, parameter :: forward=6      ! CTRLF move cursor one step right
    integer, parameter :: HELP=8         ! CTRLH give coordinates and update
    integer, parameter :: TAB=9          ! CTRLI end of input
    integer, parameter :: ctrlk=11       ! CTRLK delete to end of line
    integer, parameter :: return=13      ! CTRLM end of input
    integer, parameter :: DEL=127        ! DEL delete char left of cursor
    integer, parameter :: mode=17        ! CTRLQ toggle insert/replace
! on MAC same as arrow UP DOWN FORWARD suck
    integer, parameter :: backspace=27   ! CTRL[
!--------------------
! UP previous history line (if any)
! DOWN and LF next history line (if any)
    integer, parameter :: CTRLP=16       ! CTRLP previous in history
!    integer, parameter :: UP=27         ! uparrow previous in history
    integer, parameter :: LF=10          ! CTRLJ next in history
!--------------------
! backspace on a MAC screen
    integer, parameter :: tbackspace=8
!-----------
```

The most important is the value for "tbackspace".

The other control characters you can change to whatever you prefer. The selection here is close to emacs style editing.

On a Mac all the arrow keys give the value "27" by getkey so unfortunately they will all be interpreted as backspace. If they give different codes on you computer you can add these as well as any other preferences you have for online editing to the bintxt subroutine.

You should keep a copy of the original metlib3.F90 until you are satisfied with the editing and do not have the cursor jumping all over the screen.

There is a small history kept by bintxt, max 20 lines that can be recalled using "ctrl-P". If you have used several "ctrl-P" you can then use "ctrl-J" to recall the history lines you already passed. It is the possibility to repeat the same command several times that I use most with this facility.

4. When you have edited the metlib3+getkey.F90 file you can rename it to just metlib3.F90 and then compile and link the whole OC software with this facility using the makefile Makefile+getkey that linkes the sequential version. To run a makefile that is not called Makefile give make -f

```
>>> On Mac that is: make -f Makefile+getkey clean
>>> On Mac that is: make -f Makefile+getkey
```

   You have to use "clean" each time you want to recompile because OC has its source code on several directories and one has to do special things to force "make" to understand that it has to recompile all.

   In the Makefile+getkey the module file with the interface m_getkey.mod and the compiled getkey.o are copied from the utilities/GETKEY/ directory.

   You can modify yourself the Makefile-parallel to include getkey. I do not like Makefiles so I have many of them.

Mave fun and make OC better!