I. Note: calling C functions from Fortran is VERY compiler dependent. As I don't have Intel Fortran compiler, I tried Linux Fortran code only with GNU Fortran compiler gfortran. So there may be some problems if you try to use ifort instead. So I recommend testing using gfortran.

II. Description of the files on Linux:
1) TimedllLinux.c - C code which gets the internet time. This file must be compiled with the Fortran code which intends to get the internet time;
2) FortranTimeClient.f90 - an example of the Fortran code which uses functions from TimedllLinux.c to get the internet time. Prints out the current year, month, day, hour, minute and seconds.

III. How to use this code: compile as follows:

```
gcc -c TimedllLinux.c
gfortran -c FortranTimeClient.f90
gfortran -o getInetTimeLinux FortranTimeClient.o TimedllLinux.o
```

Here getInetTimeLinux is the name of the executable. Run the executable and see the output, which should look like this:

```
year:          13
 month:          8
 day:          4
 hour:          14
 minute:          34
 second:          1
```

Sometimes several tries are required to get the results, and it may take some time to get the output (especially on Linux, it may be about several tens of seconds).

IV. Code description. The usage of the C code differs on Windows and Linux.

On Linux there are 6 subroutines with self-descriptive names:

```
getyear(int );
getmonth(int );
getday(int );
gethour(int );
getminute(int );
getsecond(int );
```

Each of these subroutines puts a year, month, day etc. into its integer argument (as output). Note that on Linux in the Fortran code one MUST use subroutines (not functions) and pass the output as the argument of these subroutines (see the example code).

V. On Windows there're 6 functions with self-descriptive names:

```
int getyear();
int getmonth();
int getday();
```

int gethour();
int getminute();
int getsecond();
Each of these functions returns a year, month, day etc. On Windows one MUST use interfaces and functions (see example code) in the Fortran code, and save variables as return values of the functions.

So when you test your code on Linux, use SUBROUTINES for getting the time values.
When preparing the code for Windows, use INTERFACES+FUNCTIONS.

VI. Other details:
When getyear() is called, the library tries to connect to one of the time servers on the internet and get a UTC time (which is -1 hour relatively to the German winter time and -2 hours relatively to the German summer time).
Then getyear returns the following:
1) if no errors, it returns a year in form of two digits, e. g. for the current year it is 13;
2) -1 if socket problem
3) -2 if cannot connect to the server
4) -3 if connection is closed
5) -4 if recv failed
6) -5 if parsing error
One has to check the return value of getyear() before using other functions.

VII. One has to test the code intensively.
Possible issues: parsing errors (when numbers start from 0, like 04), connection errors (when no inet connection found).