TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing (ZIH)

First Steps with the GNU Debugger

HRSK Practical on Debugging, 03.04.2009

Zellescher Weg 12

Willers-Bau A106

Tel. +49 351 - 463 - 31945

Matthias Lieber (matthias.lieber@tu-dresden.de)

Tobias Hilbrich (tobias.hilbrich@zih.tu-dresden.de)

ZIH

Center for Information Services & High Performance Computing

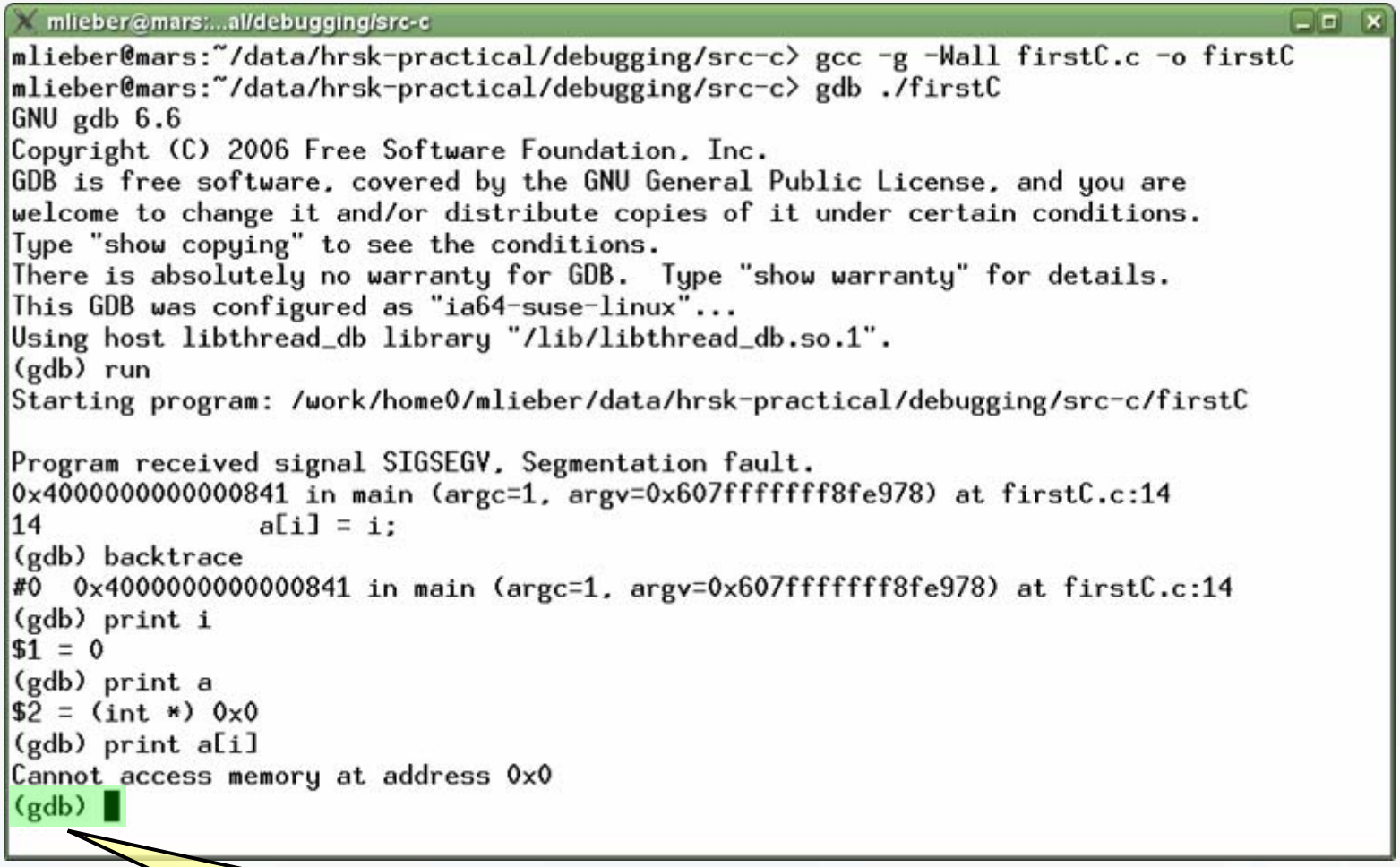
Content

- Introduction
- Main Features of the GNU Debugger
  - Run a program
  - Breakpoints
  - Examine the stack
  - Examine variables
  - Attach to a running program
  - Working with core dumps
- Exercise

The GNU Debugger (GDB)

- Free debugger
- Available for many operating systems and architectures
- De-facto standard in Linux
- No graphical user interface, but used as backend in many GUI debuggers
- Supports many programming languages including C, C++, and Fortran
- Supports multi-threaded programs using pthreads and OpenMP
- http://www.gnu.org/software/gdb/

GDB User Interface



GDB command prompt (like a Unix shell)

GDB Help

- Type **help** at the GDB command prompt
- Type **help <command>** to get help on a specific command

```
mlieber@mars:~/debug <2>
(gdb) help
List of classes of commands:

aliases -- Aliases of other commands
breakpoints -- Making program stop at certain points
data -- Examining data
files -- Specifying and examining files
internals -- Maintenance commands
obscure -- Obscure features
running -- Running the program
stack -- Examining the stack
status -- Status inquiries
support -- Support facilities
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands

Type "help" followed by a class name for a list of commands in that class.
Type "help all" for the list of all commands.
Type "help" followed by command name for full documentation.
Type "apropos word" to search for commands related to "word".
Command name abbreviations are allowed if unambiguous.
(gdb)
```

Running Programs under GDB

- Compile with **–g** compiler flag to add symbol names to the program
- Run GDB with the executable name as command-line parameter
- Enter the GDB command **run** or just **r**

```
mlieber@mars:~/debug
mlieber@mars:~/debug> icc -g firstC.c -o firstC
mlieber@mars:~/debug> gdb firstC
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "ia64-suse-linux"...
Using host libthread_db library "/lib/libthread_db.so.1".
(gdb) run
```

Control Program Execution under GDB

- Stop the program
  - Program crashes - GDB stops program just before termination
  - You hit **CTRL+C** to interrupt the program
  - Program reaches breakpoint or watchpoint
- Continue program execution
  - **c, continue** – continue running the program
  - **s, step** – execute next line of code, step into function calls
  - **n, next** – execute next line of code, but step over function calls
  - **finish** – run until program leaves current function
- Terminate the program
  - **kill** – terminate the program
  - **quit** – leave GDB

Program Crash under GDB

```
mlieber@mars:~/debug
mlieber@mars:~/debug> gdb ./exampleC
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "ia64-suse-linux"...
Using host libthread_db library "/lib/libthread_db.so.1".
(gdb) run
Starting program: /work/home0/mlieber/data/hrsk-practical/debugging/src-c/exampleC

Program received signal SIGSEGV, Segmentation fault.
0x4000000000001550 in heatInitialize (grid=0x607fffffff8de430) at exampleC-02.c:71
71                  grid->theta[i][j] = 0.0;
(gdb)
```

Breakpoints with GDB

- **b, break** – set a breakpoint
  - **break [<file>]:<line>**
  - **break [<file>]:<function>**
- **info break** – view breakpoints
- **disable** / **enable** – turn a breakpoint off/on
- **delete** – remove a breakpoint

- **break <...> if <expression>** – set a conditional breakpoint, e.g.:
  - **break example.c:123 if variable>65**
  - **break example.F90:456 if variable==1/x**

Breakpoints with GDB

```
mlieber@mars:~/debug
(gdb) break exampleF.F90:292 if dthetamax < 0.05
Breakpoint 1 at 0x4000000000000a790: file exampleF.F90, line 292.
(gdb) info break
Num Type           Disp Enb Address            What
1   breakpoint     keep y   0x4000000000000a790 in heatexample at exampleF.F90:292
        stop only if dthetamax < 0.0499999999999999996
(gdb) run
Starting program: /work/home0/mlieber/data/hrsk-practical/debugging/src-f90/exampleF
 intial grid:
================================================================================
  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  .   .  0.9 1.3 1.4 1.3 0.9  .   .   .   .   .   .   .   .   .   .   .   .   .
  .  1.2 1.7 1.9 2.0 1.9 1.7 1.2  .   .   .   .   .   .   .   .   .   .   .   .
 0.9 1.7 2.0 2.0 2.0 2.0 2.0 1.7 0.9  .   .   .   .   .   .   .   .   .   .   .
 1.3 1.9 2.0 1.9 1.9 1.9 2.0 1.9 1.3  .   .   .   .   .   .   .   .   .   .   .
 1.4 2.0 2.0 1.9 2.0 1.9 2.0 2.0 1.4  .   .   .   .   .   .   .   .   .   .   .
 1.3 1.9 2.0 1.9 1.9 1.9 2.0 1.9 1.3  .   .   .   .   .   .   .   .   .   .   .
 0.9 1.7 2.0 2.0 2.0 2.0 2.0 1.7 0.9  .   .   .   .   .   .   .   .   .   .   .
  .  1.2 1.7 1.9 2.0 1.9 1.7 1.2  .   .   .   .   .   .   .   .   .   .   .   .
  .   .  0.9 1.3 1.4 1.3 0.9  .   .   .   .   .   .   .   .   .   .   .   .   .
  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .

Breakpoint 1, heatexample () at exampleF.F90:292
292            step = step + 1
Current language:  auto; currently fortran
(gdb) print dthetamax
$1 = 0.049864621776055544
(gdb)
```

Examine the Stack with GDB

- **bt, backtrace, where**

```
mlieber@mars:~/debug
mlieber@mars:~/debug> gdb ./exampleC
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "ia64-suse-linux"...
Using host libthread_db library "/lib/libthread_db.so.1".
(gdb) break heatInitFunc
Breakpoint 1 at 0x4000000000002020: file exampleC.c, line 96.
(gdb) run
Starting program: /work/home0/mlieber/data/hrsk-practical/debugging/src-c/exampleC

Breakpoint 1, heatInitFunc (x=0.89442719099991597) at exampleC.c:96
96          return (-4.0*x*x*x + 4.0*x*x - 1.0*x + 1.0);
(gdb) backtrace
#0  heatInitFunc (x=0.89442719099991597) at exampleC.c:96
#1  0x4000000000001ec0 in heatInitialize (grid=0x607fffffff3ea430) at exampleC.c:85
#2  0x4000000000039a0 in main (argc=1, argv=0x607fffffff3ea728) at exampleC.c:255
(gdb)
```

Stack frame number

Examine Variables with GDB

- Stop the program
- Debugger can show
  - Local variables visible in the current stack frame
  - Global variables
- Select a stack frame
  - **up / down** – walk up / down the stack
  - **frame [<number>]** – select or show stack frame
- Print variables
  - **p, print <variable>** - print variable value
  - Also works for expressions, e.g.: **print mygrid.theta[i+1][j]**
  - **ptype <variable>** - print type of variable

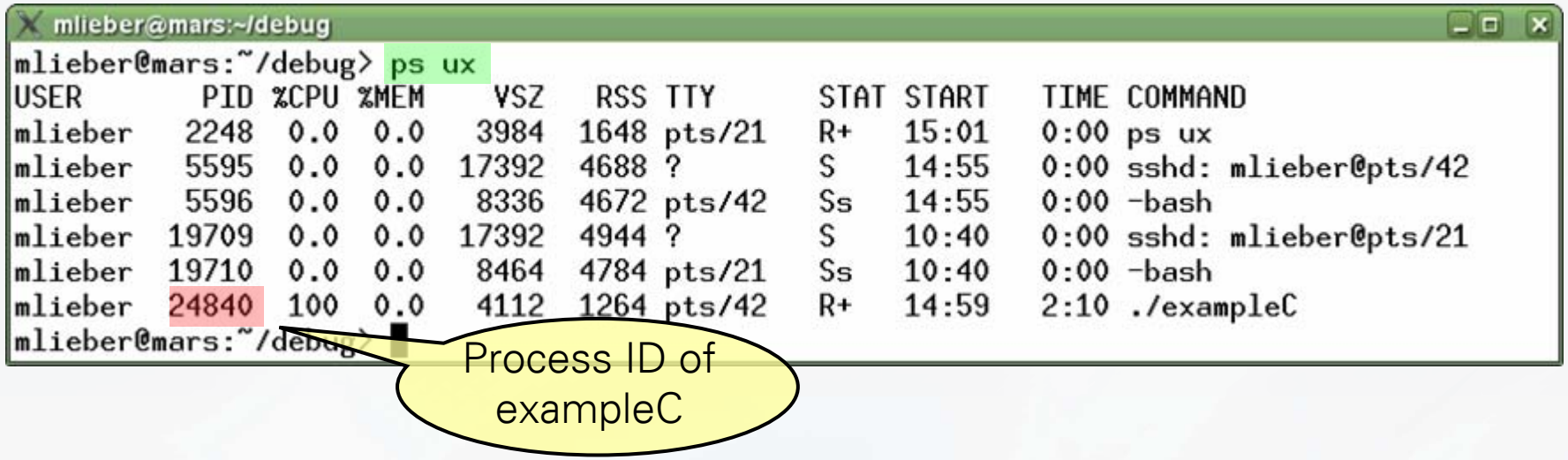
Examine Variables with GDB

```
mlieber@mars:~/debug
(gdb) break heatInitFunc
Breakpoint 1 at 0x4000000000002020: file exampleC.c, line 96.
(gdb) run
Starting program: /work/home0/mlieber/data/hrsk-practical/debugging/src-c/exampleC

Breakpoint 1, heatInitFunc (x=0.89442719099991597) at exampleC.c:96
96          return (-4.0*x*x*x + 4.0*x*x - 1.0*x + 1.0);
(gdb) backtrace
#0  heatInitFunc (x=0.89442719099991597) at exampleC.c:96
#1  0x4000000000001ec0 in heatInitialize (grid=0x607fffffefae430) at exampleC.c:85
#2  0x40000000000039a0 in main (argc=1, argv=0x607fffffefae728) at exampleC.c:255
(gdb) frame 2
#2  0x40000000000039a0 in main (argc=1, argv=0x607fffffefae728) at exampleC.c:255
255         heatInitialize(&mygrid);
(gdb) print step
$1 = 536870912
(gdb) print mygrid
$2 = {theta = 0x6000000000008010, thetanew = 0x60000000000080d0, xsize = 20, ysize = 20,
  dx = 1, dy = 1, k = 1}
(gdb) ptype mygrid
type = struct {
    double **theta;
    double **thetanew;
    int xsize;
    int ysize;
    double dx;
    double dy;
    double k;
}
(gdb)
```

Attach to a Running Program with GDB

- Useful if program runs long time and you want to check if it is doing something wrong
- On a HPC cluster: login to the node where your program is running
- Find out the process ID (PID) of your program with **ps** or **top**



- Attach GDB to the running process
  - **gdb –pid 24840**
  - Stops process

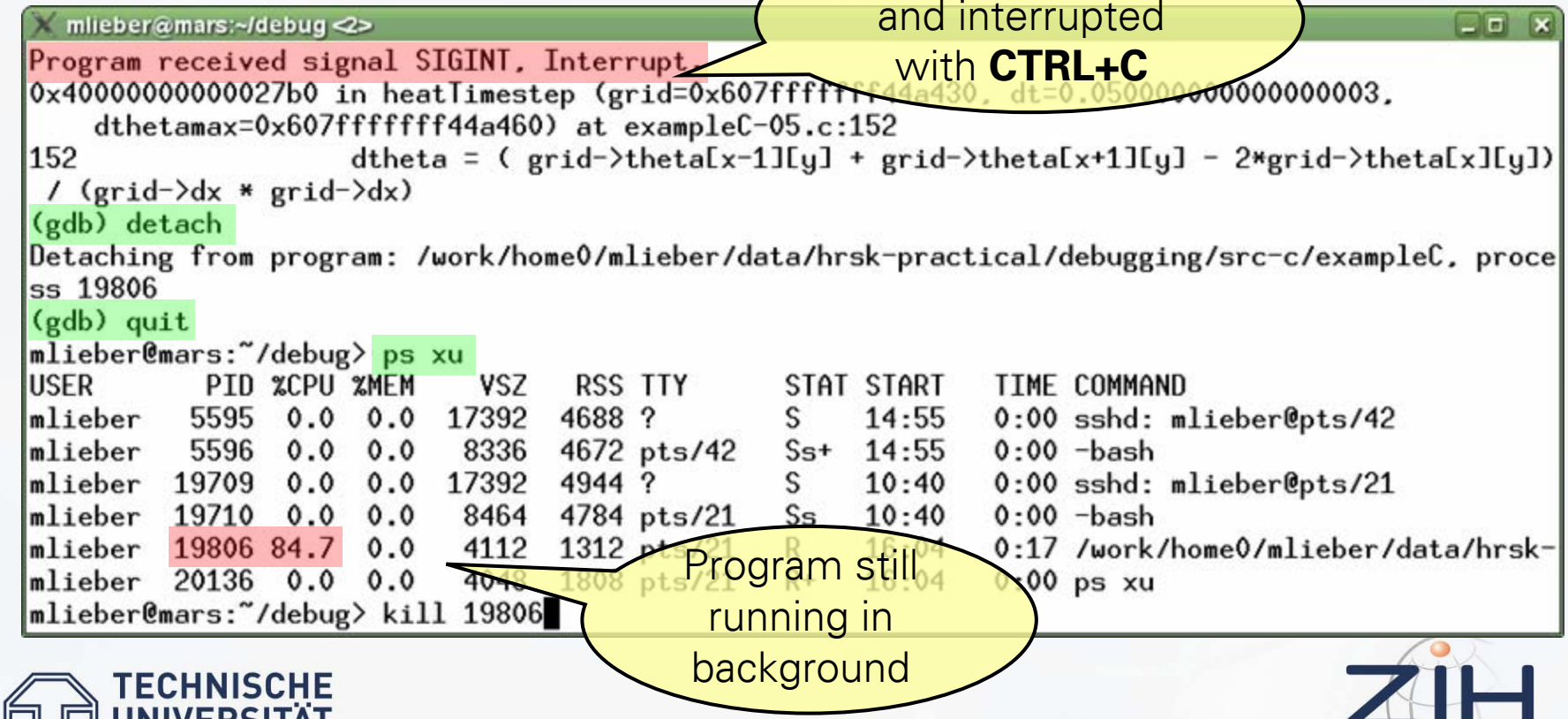
Attach to a Running Program with GDB

```
mlieber@mars:~/debug <2>
mlieber@mars:~/debug> gdb -pid 24840
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "ia64-suse-linux".
Attaching to process 24840
Reading symbols from /work/home0/mlieber/data/hrsk-practical/debugging/src-c/exampleC...done.
Using host libthread_db library "/lib/libthread_db.so.1".
Reading symbols from /lib/libm.so.6.1...done.
Loaded symbols for /lib/libm.so.6.1
Reading symbols from /lib/libc.so.6.1...done.
Loaded symbols for /lib/libc.so.6.1
Reading symbols from /lib/ld-linux-ia64.so.2...done.
Loaded symbols for /lib/ld-linux-ia64.so.2
0x4000000000002be1 in heatTimestep (grid=0x607fffffffcb2700, dt=0.050000000000000003,
    dthetamax=0x607fffffffcb2730) at exampleC-05.c:154
154                 grid->thetanew[x][y] = grid->theta[x][y] + grid->k * dtheta * dt;
(gdb) backtrace
#0  0x4000000000002be1 in heatTimestep (grid=0x607fffffffcb2700, dt=0.050000000000000003,
    dthetamax=0x607fffffffcb2730) at exampleC-05.c:154
#1  0x4000000000003ae0 in main (argc=1, argv=0x607fffffffcb29f8) at exampleC-05.c:275
(gdb)
```

Stopped program at this source code line

Detach a Program from GDB

- You can detach from a program which
  - was attached to GDB
  - was started under GDB – puts the process in the background
- GDB command: **detach**



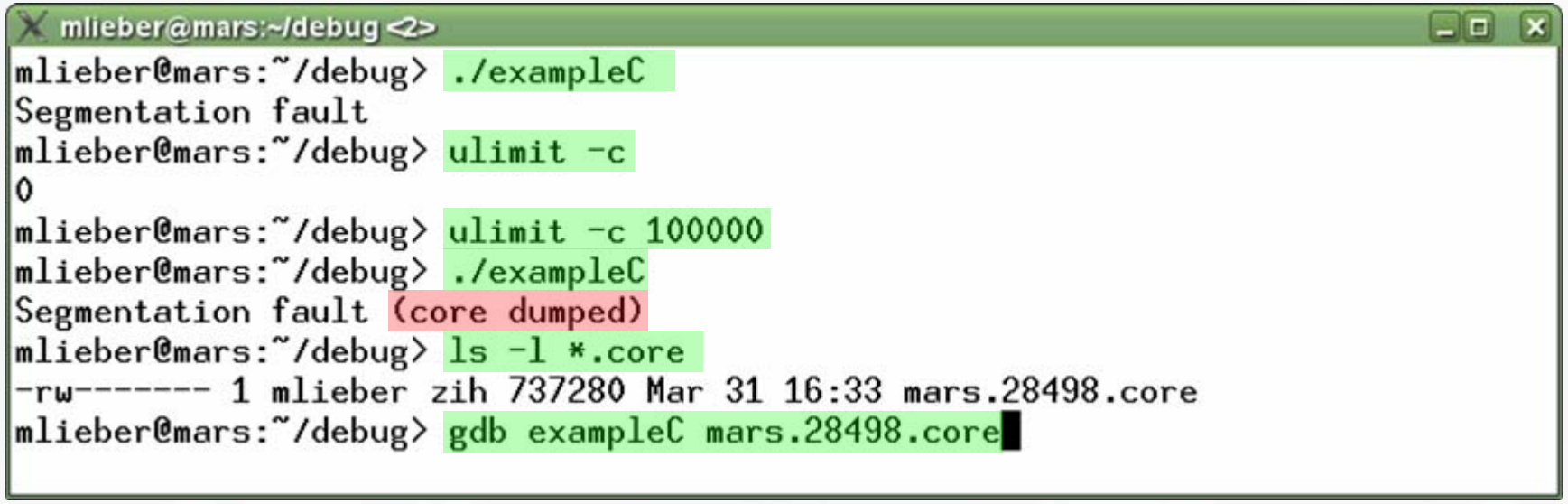
Working with Core Dumps

- Core dumps are memory and register state of a crashed program written to disk
- Check current core dump limit (reports kB): **ulimit –c**
- If necessary, set the limit for core dumps:

  ulimit –c 100000 (set limit to 100MB)
- Intel Fortran requires this environment variable to be set before running the program:

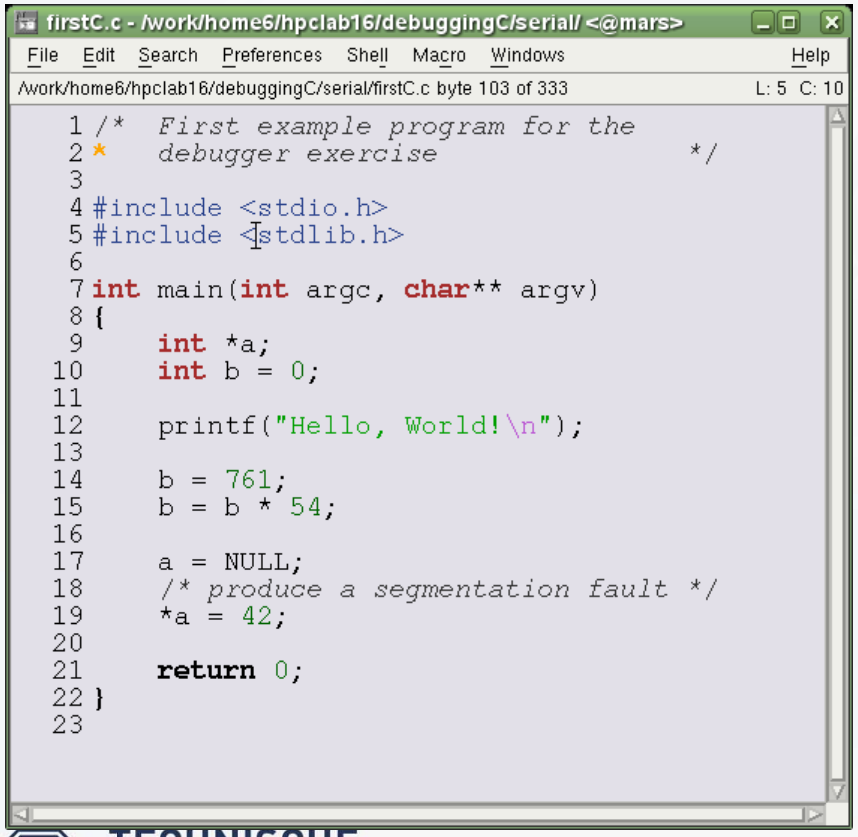
  export decfort_dump_flag=yes (Intel Fortran only)
- Start the program without debugger: **./exampleC**
- When the program crashes, core dump(s) **[HOSTNAME].[PID].core** will be created
- Analyze the core dump with GDB: **gdb <executable> <corefile>**

Working with Core Dumps

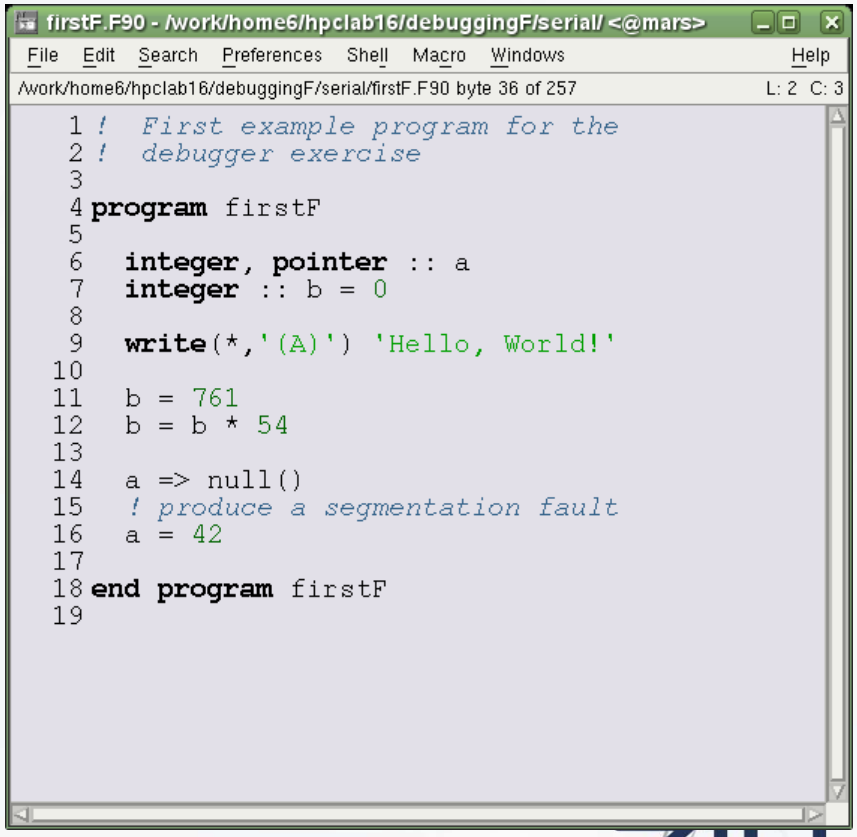


Exercise 1: Segmentation Fault

- Login on **mars**
- **cd debuggingC/serial/**
- Open **firstC.c**



- Login on **mars**
- **cd debuggingF/serial/**
- Open **firstF.F90**



Exercise 1: Segmentation Fault

Compile:

- **icc –g –O0 firstC.c –o first.exe**

Run:

- **./first.exe**

Debug:

- **gdb ./first.exe**
- Try also core file debugging

More to try out:

- Compile without **-g**
- Compile with **–O2** instead of **–O0**
- Compile with **gcc** instead of **icc**

Compile:

- **ifort –g –O0 firstF.F90 –o first.exe**

Run:

- **./first.exe**

Debug:

- **gdb ./first.exe**
- Try also core file debugging

More to try out:

- Compile without **-g**
- Compile with **–O2** instead of **–O0**
- Compile with **gfortran** instead of **ifort**

GDB commands to use: break, run, step, continue, ptype, print, info break, kill, quit

Exercise 2: Playing with the Heat Equation Program

- Compile **exampleC.c**
- Run under GDB

- Compile **exampleF.F90**
- Run under GDB
- Hint: use DDT's GDB, which has a better support for F90
  - **module load ddt**

- Set a breakpoint at the first line of the main program
  - Play with the commands **step, next, finish, backtrace**
  - Find out the exact value of **engerylnitial** using GDB
- Use a conditional breakpoint to find out the value of **dthetamax** after **step** 100 has been computed
- Find out in which iteration **step** the grid value at **theta(15,10)** becomes larger than 0

Exercise 3: Attach to a Running Program

- Compile **exampleC-05.c**
- Compile **exampleF-05.F90**
- Run without GDB
  - Program hangs
  - Kill with **CTRL+C**
- Open an xterm and start the program in the new terminal window
  - **xterm &**
- Attach GDB to the program
  - Run **ps ux** or **top** to find out the PID
  - **gdb –pid <PID>**
- What is the problem with this program? Print the iteration step and compare with the number of steps the correct program takes.

GDB commands to use: continue, CTRL+C, backtrace, up/down, print, etc.