# Embedded Software

## Debugging

# Agenda

- Debugging
- The art of debugging using gdb
- Other tools

# Debugging (& testing)

- Why is it a big challenge finding errors?
- The different types of programming errors
- The process of finding errors via Debugging

- Proactively reduce errors

# Why is it a big challenge finding errors?

- The bug is rare and seemingly non-deterministic - Heisenbug
  - Thread related
    - Timing related manifested as -> wrong incoming events, invalid syncing, data not there anymore, deadlocks
  - Incorrect handling of data resulting in corrupt memory - Symptom and Cause not related
  - External causes
  - They may not happen for long periods of time
  - Incorrect assumptions on data leading to faulty logic

# The different types of programming errors

- Syntax and type handling
- Design in terms of algorithms
- Logic errors
  - if, switch, loop constructs...
- Memory errors
  - Null pointers, array bound, stray pointers & leaks
- Interfaces
  - Communication between parts - classes, modules, programs
- Extraneous
  - Hardware failure, software failure in other systems
- Threads and synchronization
  - Deadlocks
  - Priority based (inversion etc.)

# The process of finding errors via Debugging

- Reproducibility
- Reduction
- Deduction
- Experimentation
- Experience
- Tenacity

# Proactively reduce errors

- Ensure you understand the premiss of the system you are working on
- Keep interfaces simple and intuitive
- Keep things simple e.g. “*One class - One responsibility*”
- Use asserts (compile or/and runtime)
- Enable compiler warnings - At max if possible
- Use “*lint*” or other tool to validate your software
- Test test test - Verify correctness via test cases
  - Write code that is testable... (next semester)
  - All levels of abstraction

# The art of debugging using gdb

- Short intro
- Debugging using gdb
- Cross debugging using gdb
- Enabling core dumps

# GDB

- GNU Debugger
  - Written by Richard Stallman in 1986 as part of the GNU System
- Classic debugger with lots of features
- Text based debugger
  - Interactive shell with auto completion
    - Everything is done this way
  - Script support - now python
  - Upcoming support for *checkpoint* semantics
- GUI Frontends
  - Numerous frontends, parsing all actual commands to the gdb program
    - ddd, eclipse, Visual Studio etc.

# GDB

- GNU Debugger
  - Written by Richard Stallman in 1986 as part of the GNU System
- Classic debugger with lots of features
- Text based debugger
  - Interactive shell with auto completion
    - Everything is done this way
  - Script support - now python
  - Upcoming support for *checkpoint* semantics
- GUI Frontends
  - Numerous frontends, parsing all actual commands to the gdb program
    - ddd, eclipse, Visual Studio etc.

```
import gdb
def print_node(value):
     frame = gdb.selected_frame()
try:
     val = gdb.Frame.read_var(frame, value)
except:
     print “No such variable”
     return
if str(val.type) == “struct _node *”:
     print “Weight: ” + str(val["weight"])
     print “Tag: ” + str(val["tag"])
else:
     print “Is not a node (” + str(val.type) + “)”
....
python print_node(“mynode”)
```

# The art of debugging using gdb

- Precondition - compilation
  - Flags  -O0 -g
    - Disable optimization and add debug information
  - *g++ -o main* ***-O0 -g*** *main.cpp*
- Why?
  - Missing symbols → Hard to debug
  - Optimization removes functions and variables, inlining etc.
    - Optimization might provoke the problem...

# Debugging using gdb

```
stud@ubuntu:/tmp% g++ -O0 -g -o main main.cpp
stud@ubuntu:/tmp% gdb ./main
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gp
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copy
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/main...done.
(gdb) b badFunc
Breakpoint 1 at 0x80486da: file main.cpp, line 5.
(gdb) r
Starting program: /tmp/main
Bad program...

Breakpoint 1, badFunc () at main.cpp:5
5         int* p = 0;
(gdb) n
6         *p = 1;
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x080486e4 in badFunc () at main.cpp:6
6         *p = 1;
(gdb) bt
#0  0x080486e4 in badFunc () at main.cpp:6
#1  0x0804871e in main (argc=1, argv=0xbffff104) at main.cpp:13
(gdb) print p
$1 = (int *) 0x0
(gdb)
```

```
#include <iostream>

void badFunc()
{
    int* p = 0;
    *p = 1;
}

int main(int argc, char* argv[])
{
    std::cout << "Bad program..."
              << std::endl;

    badFunc();

    return 0;
}
```

# Debugging using gdb

```
stud@ubuntu:/tmp% g++ -O0 -g -o main main.cpp
stud@ubuntu:/tmp% gdb ./main
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gp
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copy
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/main...done.
(gdb) b badFunc
Breakpoint 1 at 0x80486da: file main.cpp, line 5.
(gdb) r
Starting program: /tmp/main
Bad program...

Breakpoint 1, badFunc () at main.cpp:5
5         int* p = 0;
(gdb) n
6         *p = 1;
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x080486e4 in badFunc () at main.cpp:6
6         *p = 1;
(gdb) bt
#0  0x080486e4 in badFunc () at main.cpp:6
#1  0x0804871e in main (argc=1, argv=0xbffff104) at main.cpp:13
(gdb) print p
$1 = (int *) 0x0
(gdb)
```

```cpp
#include <iostream>

void badFunc()
{
    int* p = 0;
    *p = 1;
}

int main(int argc, char* argv[])
{
    std::cout << "Bad program..."
              << std::endl;

    badFunc();

    return 0;
}
```

Compile

# Debugging using gdb

```
stud@ubuntu:/tmp% g++ -O0 -g -o main main.cpp
stud@ubuntu:/tmp% gdb ./main
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gp
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copy
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/main...done.
(gdb) b badFunc
Breakpoint 1 at 0x80486da: file main.cpp, line 5.
(gdb) r
Starting program: /tmp/main
Bad program...

Breakpoint 1, badFunc () at main.cpp:5
5         int* p = 0;
(gdb) n
6         *p = 1;
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x080486e4 in badFunc () at main.cpp:6
6         *p = 1;
(gdb) bt
#0  0x080486e4 in badFunc () at main.cpp:6
#1  0x0804871e in main (argc=1, argv=0xbffff104) at main.cpp:13
(gdb) print p
$1 = (int *) 0x0
(gdb)
```

```
#include <iostream>

void badFunc()
{
    int* p = 0;
    *p = 1;
}

int main(int argc, char* argv[])
{
    std::cout << "Bad program..."
              << std::endl;

    badFunc();

    return 0;
}
```

Start gdb with program



ENGINEERING COLLEGE OF AARHUS

# Debugging using gdb

```
stud@ubuntu:/tmp% g++ -O0 -g -o main main.cpp
stud@ubuntu:/tmp% gdb ./main
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gp
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copy
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/main...done.
(gdb) b badFunc
Breakpoint 1 at 0x80486da: file main.cpp, line 5.
(gdb) r
Starting program: /tmp/main
Bad program...

Breakpoint 1, badFunc () at main.cpp:5
5         int* p = 0;
(gdb) n
6         *p = 1;
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x080486e4 in badFunc () at main.cpp:6
6         *p = 1;
(gdb) bt
#0  0x080486e4 in badFunc () at main.cpp:6
#1  0x0804871e in main (argc=1, argv=0xbffff104) at main.cpp:13
(gdb) print p
$1 = (int *) 0x0
(gdb)
```

```
#include <iostream>

void badFunc()
{
    int* p = 0;
    *p = 1;
}

int main(int argc, char* argv[])
{
    std::cout << "Bad program..."
              << std::endl;

    badFunc();

    return 0;
}
```

Add breakpoint at badFunc

# Debugging using gdb

```
stud@ubuntu:/tmp% g++ -O0 -g -o main main.cpp
stud@ubuntu:/tmp% gdb ./main
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gp
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copy
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/main...done.
(gdb) b badFunc
Breakpoint 1 at 0x80486da: file main.cpp, line 5.
(gdb) r
Starting program: /tmp/main
Bad program...

Breakpoint 1, badFunc () at main.cpp:5
5         int* p = 0;
(gdb) n
6         *p = 1;
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x080486e4 in badFunc () at main.cpp:6
6         *p = 1;
(gdb) bt
#0  0x080486e4 in badFunc () at main.cpp:6
#1  0x0804871e in main (argc=1, argv=0xbffff104) at main.cpp:13
(gdb) print p
$1 = (int *) 0x0
(gdb)
```

```
#include <iostream>

void badFunc()
{
    int* p = 0;
    *p = 1;
}

int main(int argc, char* argv[])
{
    std::cout << "Bad program..."
              << std::endl;

    badFunc();

    return 0;
}
```

Run program

# Debugging using gdb

```
stud@ubuntu:/tmp% g++ -O0 -g -o main main.cpp
stud@ubuntu:/tmp% gdb ./main
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gp
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copy
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/main...done.
(gdb) b badFunc
Breakpoint 1 at 0x80486da: file main.cpp, line 5.
(gdb) r
Starting program: /tmp/main
Bad program...

Breakpoint 1, badFunc () at main.cpp:5
5         int* p = 0;
(gdb) n
6         *p = 1;
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x080486e4 in badFunc () at main.cpp:6
6         *p = 1;
(gdb) bt
#0  0x080486e4 in badFunc () at main.cpp:6
#1  0x0804871e in main (argc=1, argv=0xbffff104) at main.cpp:13
(gdb) print p
$1 = (int *) 0x0
(gdb)
```

```
#include <iostream>

void badFunc()
{
    int* p = 0;
    *p = 1;
}

int main(int argc, char* argv[])
{
    std::cout << "Bad program..."
              << std::endl;

    badFunc();

    return 0;
}
```

Hit breakpoint

# Debugging using gdb

```
stud@ubuntu:/tmp% g++ -O0 -g -o main main.cpp
stud@ubuntu:/tmp% gdb ./main
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gp
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copy
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/main...done.
(gdb) b badFunc
Breakpoint 1 at 0x80486da: file main.cpp, line 5.
(gdb) r
Starting program: /tmp/main
Bad program...

Breakpoint 1, badFunc () at main.cpp:5
5           int* p = 0;
(gdb) n
6           *p = 1;
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x080486e4 in badFunc () at main.cpp:6
6           *p = 1;
(gdb) bt
#0  0x080486e4 in badFunc () at main.cpp:6
#1  0x0804871e in main (argc=1, argv=0xbffff104) at main.cpp:13
(gdb) print p
$1 = (int *) 0x0
(gdb)
```

```
#include <iostream>

void badFunc()
{
    int* p = 0;
    *p = 1;
}

int main(int argc, char* argv[])
{
    std::cout << "Bad program..."
              << std::endl;

    badFunc();

    return 0;
}
```

Single stepping through code

# Debugging using gdb

```
stud@ubuntu:/tmp% g++ -O0 -g -o main main.cpp
stud@ubuntu:/tmp% gdb ./main
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gp
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copy
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/main...done.
(gdb) b badFunc
Breakpoint 1 at 0x80486da: file main.cpp, line 5.
(gdb) r
Starting program: /tmp/main
Bad program...

Breakpoint 1, badFunc () at main.cpp:5
5         int* p = 0;
(gdb) n
6         *p = 1;
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x080486e4 in badFunc () at main.cpp:6
6         *p = 1;
(gdb) bt
#0  0x080486e4 in badFunc () at main.cpp:6
#1  0x0804871e in main (argc=1, argv=0xbffff104) at main.cpp:13
(gdb) print p
$1 = (int *) 0x0
(gdb)
```

```cpp
#include <iostream>

void badFunc()
{
    int* p = 0;
    *p = 1;
}

int main(int argc, char* argv[])
{
    std::cout << "Bad program..."
              << std::endl;

    badFunc();

    return 0;
}
```

Encountered segmentation fault

# Debugging using gdb

```
stud@ubuntu:/tmp% g++ -O0 -g -o main main.cpp
stud@ubuntu:/tmp% gdb ./main
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gp
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copy
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/main...done.
(gdb) b badFunc
Breakpoint 1 at 0x80486da: file main.cpp, line 5.
(gdb) r
Starting program: /tmp/main
Bad program...

Breakpoint 1, badFunc () at main.cpp:5
5         int* p = 0;
(gdb) n
6         *p = 1;
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x080486e4 in badFunc () at main.cpp:6
6         *p = 1;
(gdb) bt
#0  0x080486e4 in badFunc () at main.cpp:6
#1  0x0804871e in main (argc=1, argv=0xbffff104) at main.cpp:13
(gdb) print p
$1 = (int *) 0x0
(gdb)
```

```
#include <iostream>

void badFunc()
{
    int* p = 0;
    *p = 1;
}

int main(int argc, char* argv[])
{
    std::cout << "Bad program..."
              << std::endl;

    badFunc();

    return 0;
}
```

Getting a backtrace

# Debugging using gdb

```
stud@ubuntu:/tmp% g++ -O0 -g -o main main.cpp
stud@ubuntu:/tmp% gdb ./main
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gp
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copy
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/main...done.
(gdb) b badFunc
Breakpoint 1 at 0x80486da: file main.cpp, line 5.
(gdb) r
Starting program: /tmp/main
Bad program...

Breakpoint 1, badFunc () at main.cpp:5
5         int* p = 0;
(gdb) n
6         *p = 1;
(gdb) n

Program received signal SIGSEGV, Segmentation fault.
0x080486e4 in badFunc () at main.cpp:6
6         *p = 1;
(gdb) bt
#0  0x080486e4 in badFunc () at main.cpp:6
#1  0x0804871e in main (argc=1, argv=0xbffff104) at main.cpp:13
(gdb) print p
$1 = (int *) 0x0
(gdb)
```

```
#include <iostream>

void badFunc()
{
    int* p = 0;
    *p = 1;
}

int main(int argc, char* argv[])
{
    std::cout << "Bad program..."
              << std::endl;

    badFunc();

    return 0;
}
```

Printing the contents of variable p

# Debugging using gdb - Demonstration time

# Debugging using ddd - Demonstration time

# Cross debugging

- Prerequisites
  - The program to be tested ready on the target
    - May be stripped - all symbols removed
  - Compiler has relevant files on the target, hack needed if distribution is *NOT* compiled with the compiler being used
  - Relevant libraries which the app may be using on target must be available on host

# Cross debugging

# Cross debugging

- Target

```
root@DevKit8000:~# gdbserver localhost:1234 ./TestTimer_d
Process ./TestTimer_d created; pid = 1306
Listening on port 1234
```

# Cross debugging

- Target

```
root@DevKit8000:~# gdbserver localhost:1234 ./TestTimer_d
Process ./TestTimer_d created; pid = 1306
Listening on port 1234
```

- Host

```
stud@ubuntu:~/repo/Development/Code/C_CPlusPlus/Libs/OSApi$ arm-none-linux-gnueabi-gdb bin/arm-linux-gcc-4.2.1/TestTimer_d
GNU gdb (CodeSourcery Sourcery G++ Lite 2007q3-51) 6.6.50.20070821-cvs
Copyright (C) 2007 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 "--host=i686-pc-linux-gnu --target=arm-none-linux-gnueabi".
For bug reporting instructions, please see:
<https://support.codesourcery.com/GNUToolchain/>.
..
(gdb) set solib-absolute-prefix /home/stud/DevKit8000Rootfs
(gdb) target remote 10.9.8.2:1234
Remote debugging using 10.9.8.2:1234
0x400007c0 in _start () from /home/stud/DevKit8000Rootfs/lib/ld-linux.so.3
(gdb) b main
Breakpoint 1 at 0x9d5c: file test/TestTimer.cpp, line 96.
(gdb) c
Continuing.
[New Thread 1308]

Breakpoint 1, main (argc=1, argv=0xbed85d64) at test/TestTimer.cpp:96
96          TestTimer t;
(gdb)
```

Start arm debugger

# Cross debugging

- Target

```
root@DevKit8000:~# gdbserver localhost:1234 ./TestTimer_d
Process ./TestTimer_d created; pid = 1306
Listening on port 1234
```

- Host

```
stud@ubuntu:~/repo/Development/Code/C_CPlusPlus/Libs/OSApi% arm-none-linux-gnueabi-gdb bin/arm-linux-gcc-4.2.1/TestTimer_d
GNU gdb (CodeSourcery Sourcery G++ Lite 2007q3-51) 6.6.50.20070821-cvs
Copyright (C) 2007 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 "--host=i686-pc-linux-gnu --target=arm-none-linux-gnueabi".
For bug reporting instructions, please see:
<https://support.codesourcery.com/GNUToolchain/>.

(gdb) set solib-absolute-prefix /home/stud/DevKit8000Rootfs
(gdb) target remote 10.9.8.2:1234
Remote debugging using 10.9.8.2:1234
0x400007c0 in _start () from /home/stud/DevKit8000Rootfs/lib/ld-linux.so.3
(gdb) b main
Breakpoint 1 at 0x9d5c: file test/TestTimer.cpp, line 96.
(gdb) c
Continuing.
[New Thread 1308]

Breakpoint 1, main (argc=1, argv=0xbed85d64) at test/TestTimer.cpp:96
96          TestTimer t;
(gdb)
```

Set shared library path for debugger
(those residing on target)

# Cross debugging

- Target

```
root@DevKit8000:~# gdbserver localhost:1234 ./TestTimer_d
Process ./TestTimer_d created; pid = 1306
Listening on port 1234
```

- Host

```
stud@ubuntu:~/repo/Development/Code/C_CPlusPlus/Libs/OSApi% arm-none-linux-gnueabi-gdb bin/arm-linux-gcc-4.2.1/TestTimer_d
GNU gdb (CodeSourcery Sourcery G++ Lite 2007q3-51) 6.6.50.20070821-cvs
Copyright (C) 2007 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 "--host=i686-pc-linux-gnu --target=arm-none-linux-gnueabi".
For bug reporting instructions, please see:
<https://support.codesourcery.com/GNUToolchain/>.
..
(gdb) set solib-absolute-prefix /home/stud/DevKit8000Rootfs
(gdb) target remote 10.9.8.2:1234
Remote debugging using 10.9.8.2:1234
0x400007c0 in _start () from /home/stud/DevKit8000Rootfs/lib/ld-linux.so.3
(gdb) b main
Breakpoint 1 at 0x9d5c: file test/TestTimer.cpp, line 96.
(gdb) c
Continuing.
[New Thread 1308]

Breakpoint 1, main (argc=1, argv=0xbed85d64) at test/TestTimer.cpp:96
96          TestTimer t;
(gdb)
```

Connect to target

# Cross debugging

- Target

```
root@DevKit8000:~# gdbserver localhost:1234 ./TestTimer_d
Process ./TestTimer_d created; pid = 1306
Listening on port 1234
```

- Host

```
stud@ubuntu:~/repo/Development/Code/C_CPlusPlus/Libs/OSApi% arm-none-linux-gnueabi-gdb bin/arm-linux-gcc-4.2.1/TestTimer_d
GNU gdb (CodeSourcery Sourcery G++ Lite 2007q3-51) 6.6.50.20070821-cvs
Copyright (C) 2007 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 "--host=i686-pc-linux-gnu --target=arm-none-linux-gnueabi".
For bug reporting instructions, please see:
<https://support.codesourcery.com/GNUToolchain/>.
..
(gdb) set solib-absolute-prefix /home/stud/DevKit8000Rootfs
(gdb) target remote 10.9.8.2:1234
Remote debugging using 10.9.8.2:1234
0x400007c0 in _start () from /home/stud/DevKit8000Rootfs/lib/ld-linux.so.3
(gdb) b main
Breakpoint 1 at 0x9d5c: file test/TestTimer.cpp, line 96.
(gdb) c
Continuing.
[New Thread 1308]

Breakpoint 1, main (argc=1, argv=0xbed85d64) at test/TestTimer.cpp:96
96          TestTimer t;
(gdb)
```

Set breakpoint on main

# Cross debugging

- Target

```
root@DevKit8000:~# gdbserver localhost:1234 ./TestTimer_d
Process ./TestTimer_d created; pid = 1306
Listening on port 1234
```

- Host

```
stud@ubuntu:~/repo/Development/Code/C_CPlusPlus/Libs/OSApi% arm-none-linux-gnueabi-gdb bin/arm-linux-gcc-4.2.1/TestTimer_d
GNU gdb (CodeSourcery Sourcery G++ Lite 2007q3-51) 6.6.50.20070821-cvs
Copyright (C) 2007 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 "--host=i686-pc-linux-gnu --target=arm-none-linux-gnueabi".
For bug reporting instructions, please see:
<https://support.codesourcery.com/GNUToolchain/>.
..
(gdb) set solib-absolute-prefix /home/stud/DevKit8000Rootfs
(gdb) target remote 10.9.8.2:1234
Remote debugging using 10.9.8.2:1234
0x400007c0 in _start () from /home/stud/DevKit8000Rootfs/lib/ld-linux.so.3
(gdb) b main
Breakpoint 1 at 0x9d5c: file test/TestTimer.cpp, line 96.
(gdb) c
Continuing.
[New Thread 1308]

Breakpoint 1, main (argc=1, argv=0xbed85d64) at test/TestTimer.cpp:96
96          TestTimer t;
(gdb)
```

Start debugging
*MUST USE Continue cmd*

# Cross debugging

- Target

```
root@DevKit8000:~# gdbserver localhost:1234 ./TestTimer_d
Process ./TestTimer_d created; pid = 1306
Listening on port 1234
```

- Host

```
stud@ubuntu:~/repo/Development/Code/C_CPlusPlus/Libs/OSApi% arm-none-linux-gnueabi-gdb bin/arm-linux-gcc-4.2.1/TestTimer_d
GNU gdb (CodeSourcery Sourcery G++ Lite 2007q3-51) 6.6.50.20070821-cvs
Copyright (C) 2007 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 "--host=i686-pc-linux-gnu --target=arm-none-linux-gnueabi".
For bug reporting instructions, please see:
<https://support.codesourcery.com/GNUToolchain/>.
..
(gdb) set solib-absolute-prefix /home/stud/DevKit8000Rootfs
(gdb) target remote 10.9.8.2:1234
Remote debugging using 10.9.8.2:1234
0x400007c0 in _start () from /home/stud/DevKit8000Rootfs/lib/ld-linux.so.3
(gdb) b main
Breakpoint 1 at 0x9d5c: file test/TestTimer.cpp, line 96.
(gdb) c
Continuing.
[New Thread 1308]

Breakpoint 1, main (argc=1, argv=0xbed85d64) at test/TestTimer.cpp:96
96          TestTimer t;
(gdb)
```

Break point reached

# Enabling core dumps

- Core dump?
  - *A core dump is file dump of the application at the time of its crash*
  - Can be loaded into gdb for inspection
  - Issue the command *ulimit -c unlimited,* to enable
    - Called prior to invoking your command

# Using Core Dumps

```
int main()
{
   int* p = NULL;
   *p = 1;
}
```

```
stud@ubuntu:/tmp% ulimit -c unlimited
stud@ubuntu:/tmp% g++ main.cpp -g -O0
stud@ubuntu:/tmp% ./a.out
[1]    4128 segmentation fault (core dumped)  ./a.out
stud@ubuntu:/tmp% gdb ./a.out core
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/a.out...done.
[New Thread 4128]

warning: Can't read pathname for load map: Input/output error.
Reading symbols from /usr/lib/libstdc++.so.6...(no debugging symbols found)...
Loaded symbols for /usr/lib/libstdc++.so.6
Reading symbols from /lib/tls/i686/cmov/libm.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libm.so.6
Reading symbols from /lib/libgcc_s.so.1...(no debugging symbols found)...done.
Loaded symbols for /lib/libgcc_s.so.1
Reading symbols from /lib/tls/i686/cmov/libc.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libc.so.6
Reading symbols from /lib/ld-linux.so.2...(no debugging symbols found)...done.
Loaded symbols for /lib/ld-linux.so.2
Core was generated by `./a.out'.
Program terminated with signal 11, Segmentation fault.
#0  0x080485a4 in main () at main.cpp:6
6         *p = 1;
(gdb)
```

# Using Core Dumps

```
int main()
{
    int* p = NULL;
    *p = 1;
}
```

SEG Fault

```
stud@ubuntu:/tmp% ulimit -c unlimited
stud@ubuntu:/tmp% g++ main.cpp -g -O0
stud@ubuntu:/tmp% ./a.out
[1]    4128 segmentation fault (core dumped)  ./a.out
stud@ubuntu:/tmp% gdb ./a.out core
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/a.out...done.
[New Thread 4128]

warning: Can't read pathname for load map: Input/output error.
Reading symbols from /usr/lib/libstdc++.so.6...(no debugging symbols found)...
Loaded symbols for /usr/lib/libstdc++.so.6
Reading symbols from /lib/tls/i686/cmov/libm.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libm.so.6
Reading symbols from /lib/libgcc_s.so.1...(no debugging symbols found)...done.
Loaded symbols for /lib/libgcc_s.so.1
Reading symbols from /lib/tls/i686/cmov/libc.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libc.so.6
Reading symbols from /lib/ld-linux.so.2...(no debugging symbols found)...done.
Loaded symbols for /lib/ld-linux.so.2
Core was generated by `./a.out'.
Program terminated with signal 11, Segmentation fault.
#0  0x080485a4 in main () at main.cpp:6
6           *p = 1;
(gdb)
```

# Using Core Dumps

```
int main()
{
    int* p = NULL;
    *p = 1;
}
```

SEG Fault

Enable core dumps

```
stud@ubuntu:/tmp% ulimit -c unlimited
stud@ubuntu:/tmp% g++ main.cpp -g -O0
stud@ubuntu:/tmp% ./a.out
[1]    4128 segmentation fault (core dumped)  ./a.out
stud@ubuntu:/tmp% gdb ./a.out core
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/a.out...done.
[New Thread 4128]

warning: Can't read pathname for load map: Input/output error.
Reading symbols from /usr/lib/libstdc++.so.6...(no debugging symbols found)...
Loaded symbols for /usr/lib/libstdc++.so.6
Reading symbols from /lib/tls/i686/cmov/libm.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libm.so.6
Reading symbols from /lib/libgcc_s.so.1...(no debugging symbols found)...done.
Loaded symbols for /lib/libgcc_s.so.1
Reading symbols from /lib/tls/i686/cmov/libc.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libc.so.6
Reading symbols from /lib/ld-linux.so.2...(no debugging symbols found)...done.
Loaded symbols for /lib/ld-linux.so.2
Core was generated by `./a.out'.
Program terminated with signal 11, Segmentation fault.
#0  0x080485a4 in main () at main.cpp:6
6           *p = 1;
(gdb)
```

# Using Core Dumps

```
int main()
{
    int* p = NULL;
    *p = 1;
}
```

SEG Fault

Enable core dumps

Compiled with debug

```
stud@ubuntu:/tmp% ulimit -c unlimited
stud@ubuntu:/tmp% g++ main.cpp -g -O0
stud@ubuntu:/tmp% ./a.out
[1]    4128 segmentation fault (core dumped)  ./a.out
stud@ubuntu:/tmp% gdb ./a.out core
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/a.out...done.
[New Thread 4128]

warning: Can't read pathname for load map: Input/output error.
Reading symbols from /usr/lib/libstdc++.so.6...(no debugging symbols found)...
Loaded symbols for /usr/lib/libstdc++.so.6
Reading symbols from /lib/tls/i686/cmov/libm.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libm.so.6
Reading symbols from /lib/libgcc_s.so.1...(no debugging symbols found)...done.
Loaded symbols for /lib/libgcc_s.so.1
Reading symbols from /lib/tls/i686/cmov/libc.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libc.so.6
Reading symbols from /lib/ld-linux.so.2...(no debugging symbols found)...done.
Loaded symbols for /lib/ld-linux.so.2
Core was generated by `./a.out'.
Program terminated with signal 11, Segmentation fault.
#0  0x080485a4 in main () at main.cpp:6
6           *p = 1;
(gdb)
```

# Using Core Dumps

```
int main()
{
    int* p = NULL;
    *p = 1;
}
```

SEG Fault

Enable core dumps

Compiled with debug

Run and core dumped

```
stud@ubuntu:/tmp% ulimit -c unlimited
stud@ubuntu:/tmp% g++ main.cpp -g -O0
stud@ubuntu:/tmp% ./a.out
[1]    4128 segmentation fault (core dumped)  ./a.out
stud@ubuntu:/tmp% gdb ./a.out core
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/a.out...done.
[New Thread 4128]

warning: Can't read pathname for load map: Input/output error.
Reading symbols from /usr/lib/libstdc++.so.6...(no debugging symbols found)...
Loaded symbols for /usr/lib/libstdc++.so.6
Reading symbols from /lib/tls/i686/cmov/libm.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libm.so.6
Reading symbols from /lib/libgcc_s.so.1...(no debugging symbols found)...done.
Loaded symbols for /lib/libgcc_s.so.1
Reading symbols from /lib/tls/i686/cmov/libc.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libc.so.6
Reading symbols from /lib/ld-linux.so.2...(no debugging symbols found)...done.
Loaded symbols for /lib/ld-linux.so.2
Core was generated by `./a.out'.
Program terminated with signal 11, Segmentation fault.
#0  0x080485a4 in main () at main.cpp:6
6           *p = 1;
(gdb)
```

# Using Core Dumps

```
int main()
{
    int* p = NULL;
    *p = 1;
}
```

SEG Fault

Enable core dumps

Compiled with debug

Run and core dumped

Loaded in gdb

```
stud@ubuntu:/tmp% ulimit -c unlimited
stud@ubuntu:/tmp% g++ main.cpp -g -O0
stud@ubuntu:/tmp% ./a.out
[1]    4128 segmentation fault (core dumped)  ./a.out
stud@ubuntu:/tmp% gdb ./a.out core
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/a.out...done.
[New Thread 4128]

warning: Can't read pathname for load map: Input/output error.
Reading symbols from /usr/lib/libstdc++.so.6...(no debugging symbols found)...
Loaded symbols for /usr/lib/libstdc++.so.6
Reading symbols from /lib/tls/i686/cmov/libm.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libm.so.6
Reading symbols from /lib/libgcc_s.so.1...(no debugging symbols found)...done.
Loaded symbols for /lib/libgcc_s.so.1
Reading symbols from /lib/tls/i686/cmov/libc.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libc.so.6
Reading symbols from /lib/ld-linux.so.2...(no debugging symbols found)...done.
Loaded symbols for /lib/ld-linux.so.2
Core was generated by `./a.out'.
Program terminated with signal 11, Segmentation fault.
#0  0x080485a4 in main () at main.cpp:6
6           *p = 1;
(gdb)
```

# Using Core Dumps

```
int main()
{
    int* p = NULL;
    *p = 1;
}
```

SEG Fault

Enable core dumps

Compiled with debug

Run and core dumped

Loaded in gdb

Error found :-)

```
stud@ubuntu:/tmp% ulimit -c unlimited
stud@ubuntu:/tmp% g++ main.cpp -g -O0
stud@ubuntu:/tmp% ./a.out
[1]    4128 segmentation fault (core dumped)  ./a.out
stud@ubuntu:/tmp% gdb ./a.out core
GNU gdb (GDB) 7.1-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /tmp/a.out...done.
[New Thread 4128]

warning: Can't read pathname for load map: Input/output error.
Reading symbols from /usr/lib/libstdc++.so.6...(no debugging symbols found)...
Loaded symbols for /usr/lib/libstdc++.so.6
Reading symbols from /lib/tls/i686/cmov/libm.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libm.so.6
Reading symbols from /lib/libgcc_s.so.1...(no debugging symbols found)...done.
Loaded symbols for /lib/libgcc_s.so.1
Reading symbols from /lib/tls/i686/cmov/libc.so.6...(no debugging symbols four
Loaded symbols for /lib/tls/i686/cmov/libc.so.6
Reading symbols from /lib/ld-linux.so.2...(no debugging symbols found)...done.
Loaded symbols for /lib/ld-linux.so.2
Core was generated by `./a.out'.
Program terminated with signal 11, Segmentation fault.
#0  0x080485a4 in main () at main.cpp:6
6           *p = 1;
(gdb)
```

IHA ENGINEERING COLLEGE OF AARHUS

# Core dumps

- Things to wary about
  - Can be very large
  - Can take considerable time to create
    - Highly depended on the media whereto they are written
  - Not a silver bullet - corrupt stack → No valid dump
- Their name can be controlled by
  - sysctl -w kernel.core_pattern = core.%e.%p

```
%p:         pid
%<NUL>:     '%' is dropped
%%:         output one '%'
%u:         uid
%g:         gid
%s:         signal number
%t:         UNIX time of dump
%h:         hostname
%e:         executable filename
%<OTHER>: both are dropped
```

# Other tools

- Valgrind, Hellgrind + Cachegrind
- efence

# Valgrind and friends

- Valgrind (memcheck)
  - Features (<url for valgrind>)
    - *Accessing memory you shouldn't, e.g. overrunning and underrunning heap blocks, overrunning the top of the stack, and accessing memory after it has been freed.*
    - *Using undefined values, i.e. values that have not been initialised, or that have been derived from other undefined values.*
    - *Incorrect freeing of heap memory, such as double-freeing heap blocks, or mismatched use of malloc/new/new[] versus free/delete/delete[]*
    - *Overlapping src and dst pointers in memcpy and related functions.*
    - *Memory leaks.*
  - Invoke
    - valgrind <program name> # Lots of options - read man page :-)

# Valgrind and friends

- Callgrind
  - Features
    - Profiling
  - Invoke
    - valgrind --tool=callgrind <program name>
  - Nice to know
    - Install and use kcachegrind

# Valgrind and friends

- Helgrind
  - Features
    - *Misuses of the POSIX pthreads API*
    - *Potential deadlocks arising from lock ordering problems*
    - *Data races -- accessing memory without adequate locking or synchronization*
  - Invoke
    - valgrind --tool=helgrind <program name>