

Introduction to Parallel Processing

Lecture 8 : Debugging MPI Applications

Professor Amanda Bienz

Debugging MPI Programs

- **MPI Programs are difficult to debug!**
- No indication which process error occurs on
- Sometimes, no error until running on 1000s of processes
- Often, error occurs during MPI communication
- Serial methods of debugging don't always work (e.g. print statements)

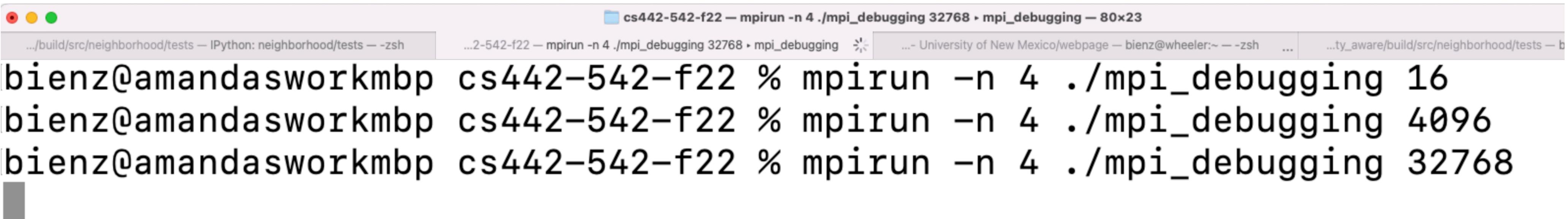
Some Common Errors

- “**Invalid rank**” :
 - Some MPI communication has a rank that is < 0 or >= num_procs

```
bienz@amandasworkmbp cs442-542-f22 % mpicxx -o mpi_debugging mpi_debugging.cpp
bienz@amandasworkmbp cs442-542-f22 % mpirun -n 4 ./mpi_debugging 32768
Abort(737823238) on node 3 (rank 3 in comm 0): Fatal error in internal_Send: Invalid rank, error stack:
internal_Send(120): MPI_Send(buf=0x7f8775784000, count=32768, MPI_DOUBLE, 4, 0,
MPI_COMM_WORLD) failed
internal_Send(78)..: Invalid rank has value 4 but must be nonnegative and less than 4
bienz@amandasworkmbp cs442-542-f22 %
```

Some Common Errors

- If your program hangs indefinitely:
 - Almost always error in communication
 - One way to debug : print out info about communication, may sometimes need to comment out communication
 - If only for large message sizes, likely MPI_Send with no matching MPI_Recv

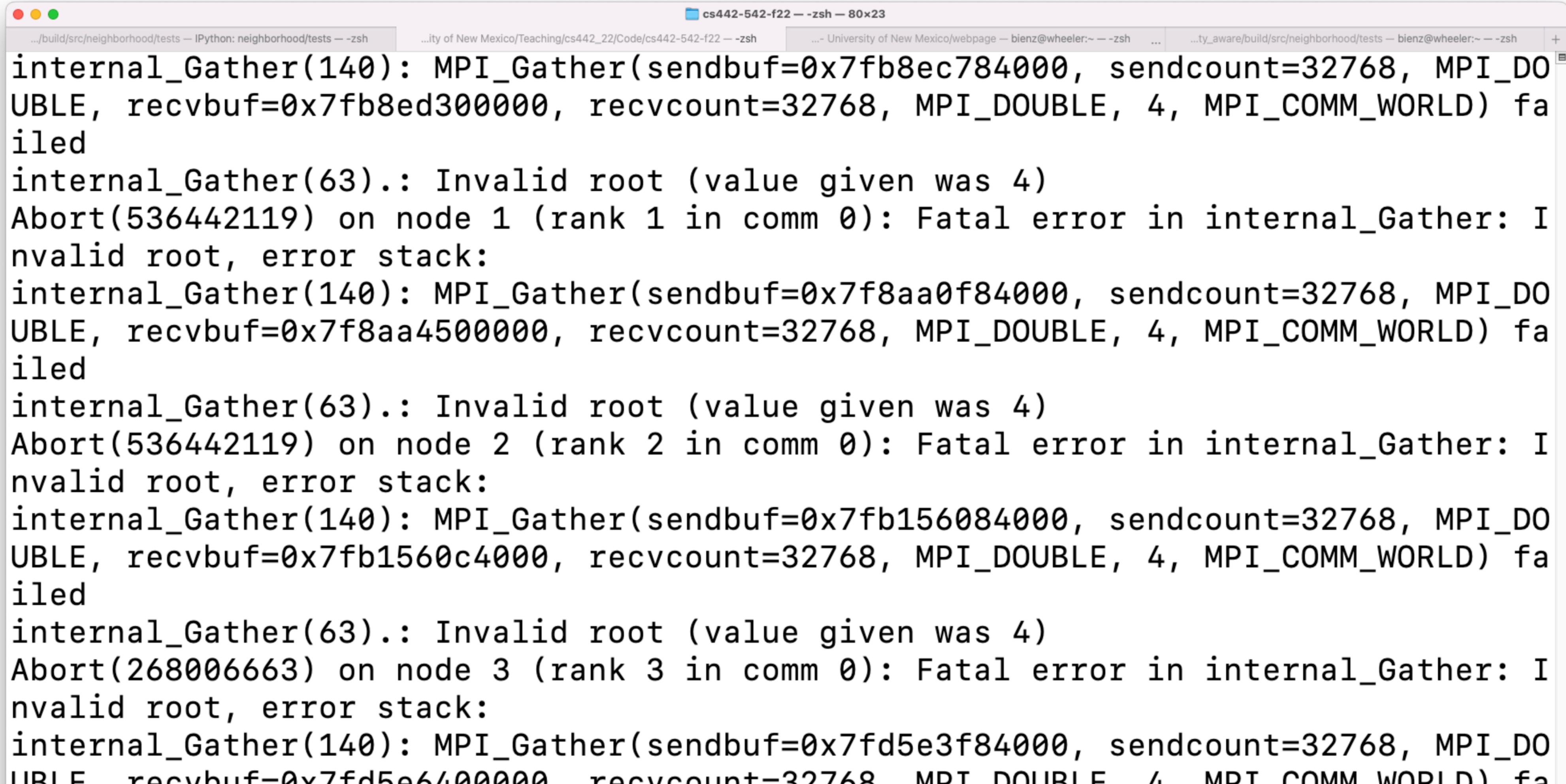


A screenshot of a terminal window titled "cs442-542-f22 — mpirun -n 4 ./mpi_debugging 32768 * mpi_debugging — 80x23". The terminal shows three lines of text output from the MPI application:

```
bienz@amandasworkmbp cs442-542-f22 % mpirun -n 4 ./mpi_debugging 16
bienz@amandasworkmbp cs442-542-f22 % mpirun -n 4 ./mpi_debugging 4096
bienz@amandasworkmbp cs442-542-f22 % mpirun -n 4 ./mpi_debugging 32768
```

Some Common Errors

- “**Invalid root**”: **root < 0 or >= num_procs**



The image shows a terminal window with multiple tabs. The active tab displays MPI error messages. The errors are repeated for four nodes (rank 1 to rank 4) and involve the MPI_Gather function. Each error message includes the internal_Gather call details (sendbuf, sendcount, MPI_DOUBLE, MPI_COMM_WORLD), followed by "failed", then "internal_Gather(63).: Invalid root (value given was 4)", and finally "Abort(536442119) on node X (rank Y in comm 0): Fatal error in internal_Gather: Invalid root, error stack".

```
internal_Gather(140): MPI_Gather(sendbuf=0x7fb8ec784000, sendcount=32768, MPI_DOUBLE, recvbuf=0x7fb8ed300000, recvcount=32768, MPI_DOUBLE, 4, MPI_COMM_WORLD) failed
internal_Gather(63).: Invalid root (value given was 4)
Abort(536442119) on node 1 (rank 1 in comm 0): Fatal error in internal_Gather: Invalid root, error stack:
internal_Gather(140): MPI_Gather(sendbuf=0x7f8aa0f84000, sendcount=32768, MPI_DOUBLE, recvbuf=0x7f8aa4500000, recvcount=32768, MPI_DOUBLE, 4, MPI_COMM_WORLD) failed
internal_Gather(63).: Invalid root (value given was 4)
Abort(536442119) on node 2 (rank 2 in comm 0): Fatal error in internal_Gather: Invalid root, error stack:
internal_Gather(140): MPI_Gather(sendbuf=0x7fb156084000, sendcount=32768, MPI_DOUBLE, recvbuf=0x7fb1560c4000, recvcount=32768, MPI_DOUBLE, 4, MPI_COMM_WORLD) failed
internal_Gather(63).: Invalid root (value given was 4)
Abort(268006663) on node 3 (rank 3 in comm 0): Fatal error in internal_Gather: Invalid root, error stack:
internal_Gather(140): MPI_Gather(sendbuf=0x7fd5e3f84000, sendcount=32768, MPI_DOUBLE, recvbuf=0x7fd5e6400000, recvcount=32768, MPI_DOUBLE, 4, MPI_COMM_WORLD) failed
```

MPI Error Handling

- You can add custom error handling
- `MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN)`
 - Default error flag : `MPI_ERRORS_ARE_FATAL`
 - `MPI_ERRORS_RETURN` : error codes are returned and program no longer crashes with error
- Can print out and handle errors on your own
 - `MPI_Error_class(error_code, &error_class)`
- Error codes : <https://www.mpi-forum.org/docs/mpi-2.2/mpi22-report/node192.htm>

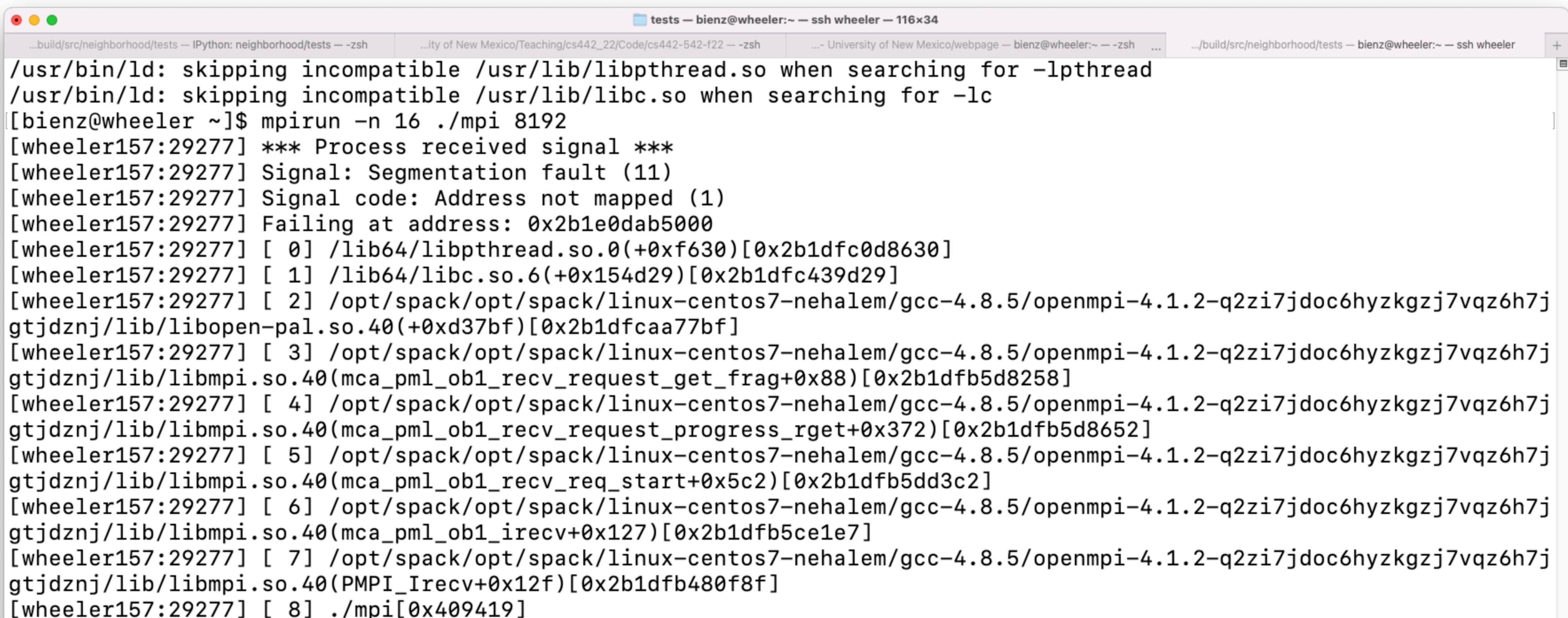
Some Common Errors

- **Segfaults : accessed unallocated memory**

```
.../build/src/neighborhood/tests — IPython: neighborhood/tests — zsh ...ity of New Mexico/Teaching/cs442_22/Code/cs442-542-f22 — zsh ...- University of New Mexico/webpage — bienz@wheeler:~ — zsh ...ty_aware/build/src/neighborhood/tests — bienz@wheeler:~ — zsh  
[bienz@amandasworkmbp cs442-542-f22 % mpicxx -o mpi_debugging mpi_debugging.cpp  
[bienz@amandasworkmbp cs442-542-f22 % mpirun -n 4 ./mpi_debugging 32768  
=====  
=====  
= BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES  
= PID 31771 RUNNING AT amandasworkmbp.unm.edu  
= EXIT CODE: 11  
= CLEANING UP REMAINING PROCESSES  
= YOU CAN IGNORE THE BELOW CLEANUP MESSAGES  
=====  
=====  
YOUR APPLICATION TERMINATED WITH THE EXIT STRING: Segmentation fault: 11 (signal 11)  
This typically refers to a problem with your application.
```

Some Common Errors

- **Segfaults : accessed unallocated memory**
 - Memory issue can occur within MPI
 - Example : changing send_buffer in MPI_Isend



The screenshot shows a terminal window with multiple tabs. The active tab displays a stack trace for a segmentation fault. The output is as follows:

```
/usr/bin/ld: skipping incompatible /usr/lib/libpthread.so when searching for -lpthread
/usr/bin/ld: skipping incompatible /usr/lib/libc.so when searching for -lc
[bienz@wheeler ~]$ mpirun -n 16 ./mpi 8192
[wheeler157:29277] *** Process received signal ***
[wheeler157:29277] Signal: Segmentation fault (11)
[wheeler157:29277] Signal code: Address not mapped (1)
[wheeler157:29277] Failing at address: 0x2b1e0dab5000
[wheeler157:29277] [ 0] /lib64/libpthread.so.0(+0xf630)[0x2b1dfc0d8630]
[wheeler157:29277] [ 1] /lib64/libc.so.6(+0x154d29)[0x2b1dfc439d29]
[wheeler157:29277] [ 2] /opt/spack/opt/spack/linux-centos7-nehalem/gcc-4.8.5/openmpi-4.1.2-q2zi7jdoc6hyzkgzj7vqz6h7jgtjdznj/lib/libopen-pal.so.40(+0xd37bf)[0x2b1dfcaa77bf]
[wheeler157:29277] [ 3] /opt/spack/opt/spack/linux-centos7-nehalem/gcc-4.8.5/openmpi-4.1.2-q2zi7jdoc6hyzkgzj7vqz6h7jgtjdznj/lib/libmpi.so.40(mca_pml_ob1_recv_request_get_frag+0x88)[0x2b1dfb5d8258]
[wheeler157:29277] [ 4] /opt/spack/opt/spack/linux-centos7-nehalem/gcc-4.8.5/openmpi-4.1.2-q2zi7jdoc6hyzkgzj7vqz6h7jgtjdznj/lib/libmpi.so.40(mca_pml_ob1_recv_request_progress_rget+0x372)[0x2b1dfb5d8652]
[wheeler157:29277] [ 5] /opt/spack/opt/spack/linux-centos7-nehalem/gcc-4.8.5/openmpi-4.1.2-q2zi7jdoc6hyzkgzj7vqz6h7jgtjdznj/lib/libmpi.so.40(mca_pml_ob1_recv_req_start+0x5c2)[0x2b1dfb5dd3c2]
[wheeler157:29277] [ 6] /opt/spack/opt/spack/linux-centos7-nehalem/gcc-4.8.5/openmpi-4.1.2-q2zi7jdoc6hyzkgzj7vqz6h7jgtjdznj/lib/libmpi.so.40(mca_pml_ob1 irecv+0x127)[0x2b1dfb5ce1e7]
[wheeler157:29277] [ 7] /opt/spack/opt/spack/linux-centos7-nehalem/gcc-4.8.5/openmpi-4.1.2-q2zi7jdoc6hyzkgzj7vqz6h7jgtjdznj/lib/libmpi.so.40(PMPI_Irecv+0x12f)[0x2b1dfb480f8f]
[wheeler157:29277] [ 8] ./mpi[0x409419]
```

Logical Bugs

- Probably the hardest to figure out
- Program doesn't crash, but you get an incorrect result
- Some ways to solve :
 - Run a bunch of unit tests
 - Check solution along the way
 - Check values before and after communication

Memory Leaks

- Valgrind!
 - This works on Wheeler
- On Mac : leaks (type ‘man leaks’ to learn more)

Some Debugging Methods

- Sophisticated tools, like Totalview
- Definitely use this if available!
- Very expensive, not on most machines

Some Debugging Methods

- GDB : The GNU Project Debugger
 - <https://www.sourcware.org/gdb/>
- Attach GDB to individual processes after they are running
- Use mpirun to launch separate instances of serial debuggers
 - `mpirun -n 4 xterm -e gdb <program_to_debug>`
 - Great for local machines, usually not a great option on supercomputers

Some Debugging Methods

- Commenting out code
 - Print statements aren't always as useful in parallel
 - If you comment out code, you know can find where the error is
- Printing information