GNU Debugger: gdb

- A text-based symbolic debugger
- Make your program stop on specified conditions.
- Examine what has happened, when your program has stopped.
- Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.

An Example Program: main.cc

```cpp
#include<iostream.h>
class Queue {
    class ListNode {
        public:
            int data;
            ListNode* next;
    };
    ListNode *front, *rear;
    public:
        Queue () {front = rear = NULL;}; ~Queue() {};
        void add (int new_item);
        void remove (int& old_front, bool& empty);
};
```
void Queue::add (int new_item) {
    ListNode* p = new ListNode;
    p->data = new_item;
    p->next = NULL;

    if (rear==NULL)
        rear = front = p;
    else
        {
            rear->next = p;
            rear = p;
        }
}

void Queue::remove (int& old_front, bool& empty) {
    empty = (front == NULL);
    if (empty)
        return;

    ListNode *p = front;
    front = front->next;
    old_front = p->data;
    delete p;
}
int main ()
{
    Queue q;
    int i;
    bool empty;

    q.add (1);
    q.add (2);
    q.remove (i,empty);
    q.remove (i,empty);
    q.add (3);
}

---

A Debugging Scenario

- g++ -g main.cc
- gdb a.out

... We now see the gdb prompt (gdb) ...

- list command: see program lines
  list 46,57 (see lines 46 – 57)

- break command: set breakpoints, the points at which we would like the program to pause
  break Queue::add (stop when entering Queue::add)
  break Queue::remove (stop when entering Queue::remove)
  break 48 (stop when reaching line 48)
run (start executing program)
... Program stops at line 18, the first line of Queue::add ...

print command: examine data
print front \rightarrow 0x0 (whose front?)
print rear \rightarrow 0x0

cont (continue) command: resume program execution
... Program execution resumes and stops again at line 18 ...

print front \rightarrow 0x20e88
print rear \rightarrow 0x20e88
print front->data \rightarrow 1
print rear->data \rightarrow ?
print front->next \rightarrow 0x0
cont
... Program execution resumes and stops again at line 34 ...

print front \rightarrow 0x20e88

print rear \rightarrow 0x20e98
print front->data \rightarrow 1
print front->next \rightarrow 0x20e98
print rear->data \rightarrow 2
print rear->next \rightarrow 0x0
cont
... Program execution resumes and stops at line 34 ...

print front \rightarrow 0x20e98
print rear \rightarrow 0x20e98
print front->next \rightarrow 0x0 print rear->data \rightarrow 2
cont
... Program execution resumes and stops at line 18 ...

- `print q.front` \(\Rightarrow 0x0\)
- `print q.rear` \(\Rightarrow 0x20e98\)
- `cont`

... Program crashes at line 26 ...

- Other useful command:
  1. `where` – show the calling stack
  2. `display` – automatically print some variables when program paused
  3. `next, step` – single step; execute the next statement and pause

- More complete information available at:
  http://www.delorie.com/gnu/docs/gdb/gdb_toc.html

- When encountering a **segmentation fault**, you probably screwed up pointers. Run your program within gdb, which will show at which line your program crashes.

  Do not assume that line is the origin of the problem; keep your mind open

- In general, when encountering bugs in your programs, use gdb; the instructor and the TA reserve the right not to answer your questions if you have not done so.