Subject: Re: How to respond when memory is exceeded
Posted by Lance on Mon, 18 Nov 2024 15:15:28 GMT
View Forum Message <> Reply to Message

No std::bad_alloc thrown. Test program killed by os.

Test on Linux 6.8.0-48-generic

```
#include <iostream>
#include <memory>
#include <format>
#include <unistd.h>

using namespace std;


unsigned long long getTotalSystemMemory()
{
    long pages = sysconf(_SC_PHYS_PAGES);
    long page_size = sysconf(_SC_PAGE_SIZE);
    return pages * page_size;
}

int main(int argc, const char *argv[])
{
 auto size = getTotalSystemMemory()/4;

 std::unique_ptr<char []> p=std::make_unique<char[]>(size);

 while(true){
  auto new_size = size *1.1;
  try{
   p = std::make_unique<char []>(new_size);
  }catch(std::bad_alloc&){
   break;
  }
  size = new_size;
  std::cout<<std::format("{} bytes allocated ok.\n", size);
 }
 std::cout<<std::format("We believe {} bytes has been successfully allocated."
  "Now test writing each byte...\n", size);


 for(size_t i=0; i< size; ++i)
  p[i] = i%256;

 std::cout<<"Program finished normally!"<<std::endl;
```

```
 return 0;
}
```

Output:

```
9190172569 bytes allocated ok.
10109189825 bytes allocated ok.
11120108807 bytes allocated ok.
12232119687 bytes allocated ok.
13455331655 bytes allocated ok.
14800864820 bytes allocated ok.
16280951302 bytes allocated ok.
17909046432 bytes allocated ok.
Killed
```