Introducing CHAP

A program to clarify dynamic memory usage in un-instrumented cores.

Tim Boddy

April 29, 2017

Background

- Was created by me in 2010 as a tool called ah64
- Was motivated by need to debug growth issues on un-instrumented cores
- Started supporting leak detection in early 2011
- Has been heavily used in our development and test life cycle for several years
- Became available as CHAP as open source under GPL-2.0 license on April 19, 2017
- http://github.com/vmware/chap

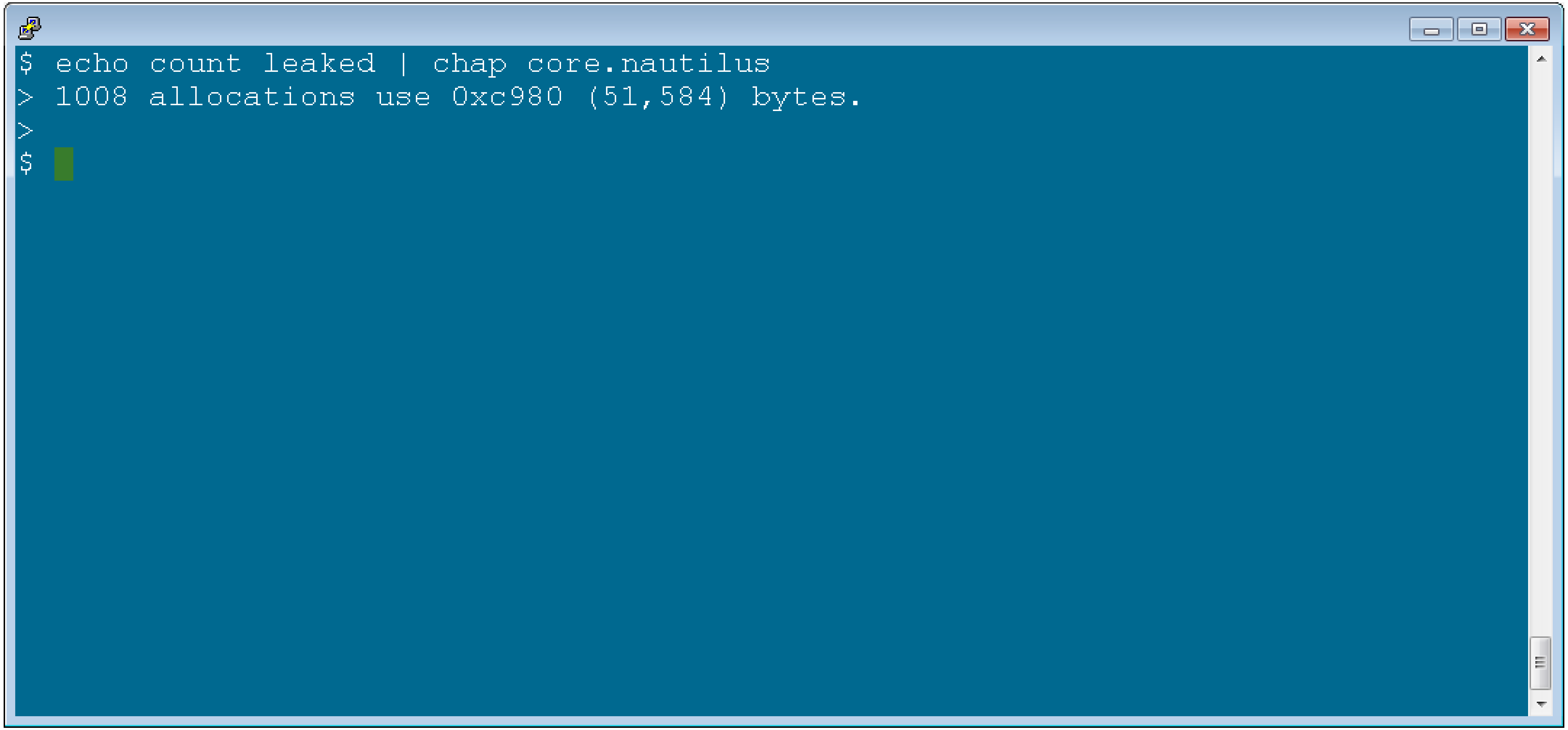
CHAP – Core Heap Analysis Program

- CHAP stands for Core Heap Analysis Program
- Reads a process image as input
  - Currently supports 32 or 64 bit ELF cores as process image
  - Does not require any advance instrumentation
- Provides information about dynamically allocated memory
  - Currently recognizes memory allocated by glibc

Some Use Cases

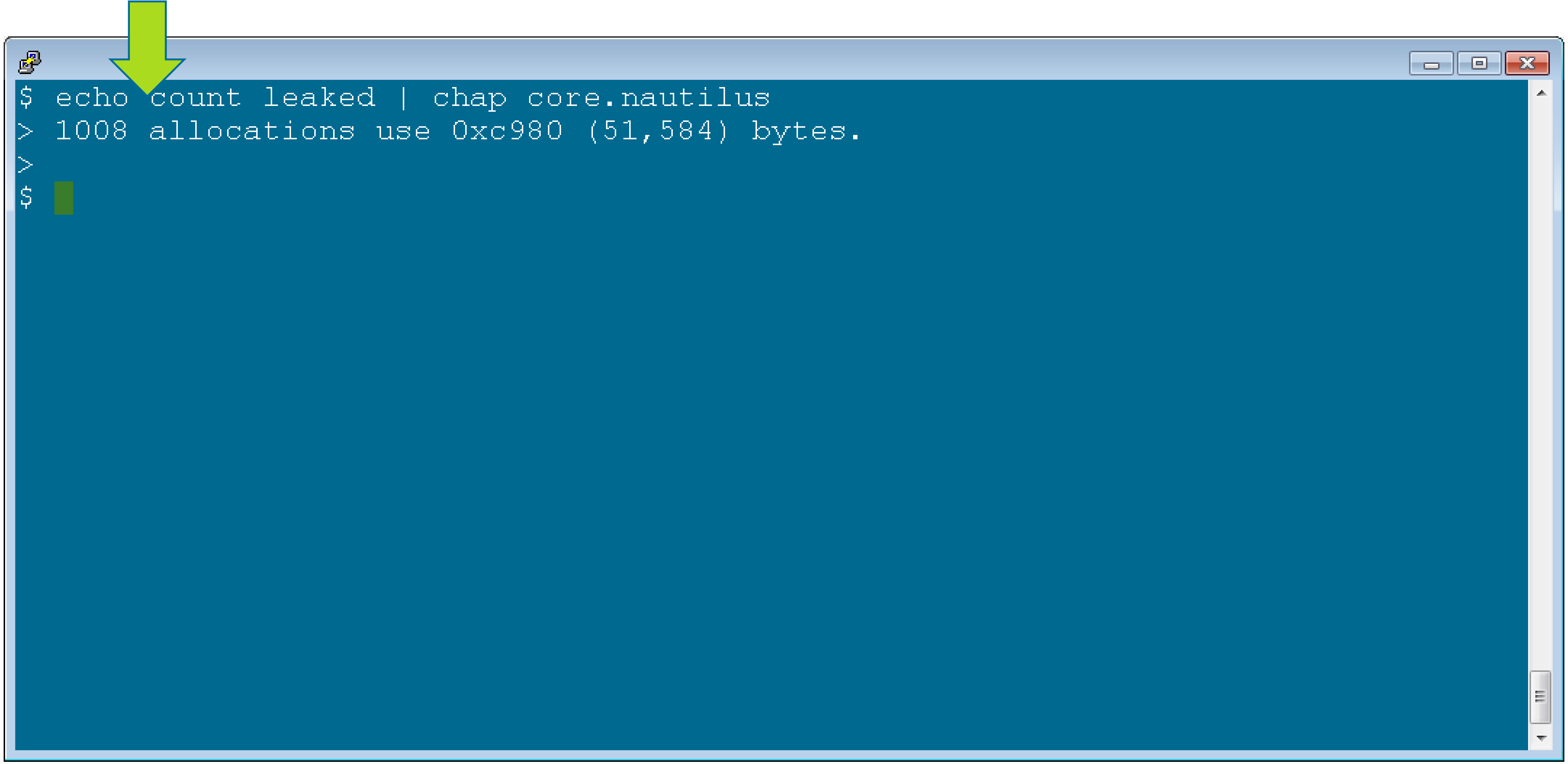
- Allows automated leak detection, even for performance tests at scale on release builds …
- Can be used interactively to do leak analysis
- Can be used interactively to do memory growth analysis
- Can automatically detect some forms of heap corruption
- Supplements debuggers such as gdb by providing status of various memory addresses

The Simplest Use Case



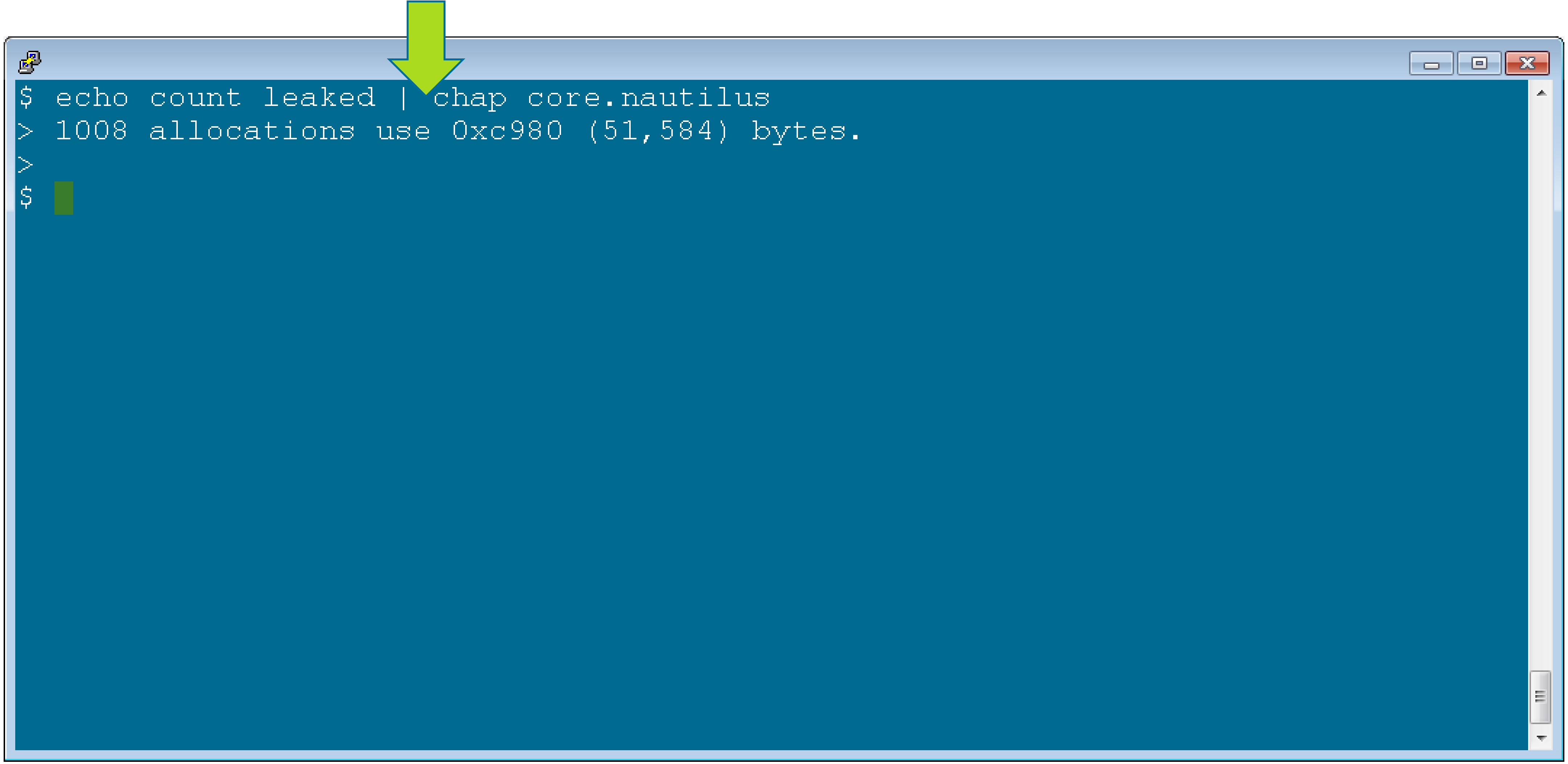
```
$ echo count leaked | chap core.nautilus
> 1008 allocations use 0xc980 (51,584) bytes.
>
$ 
```

The Simplest Use Case



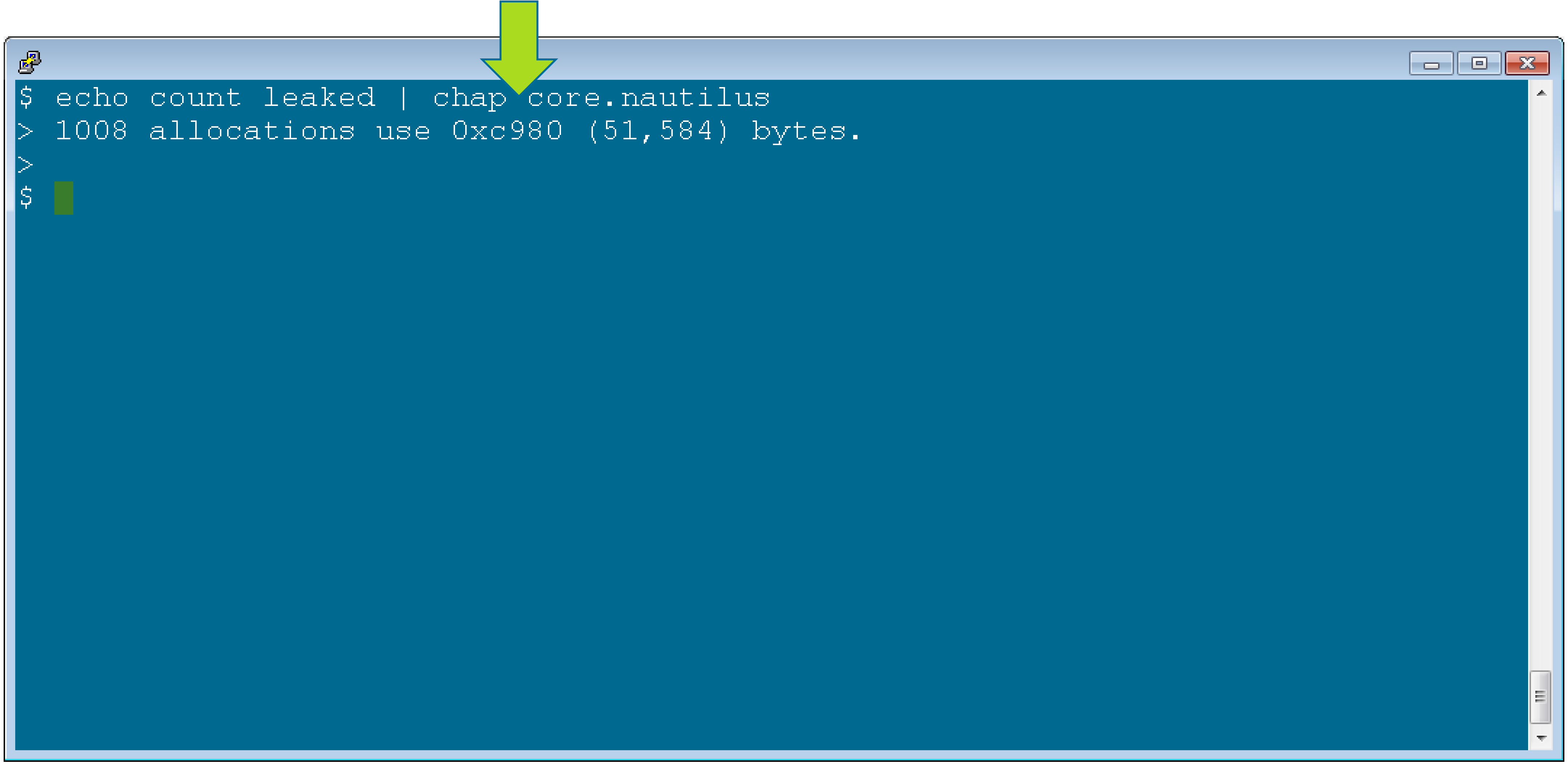
```
$ echo count leaked | chap core.nautilus
> 1008 allocations use 0xc980 (51,584) bytes.
>
$
```

The Simplest Use Case



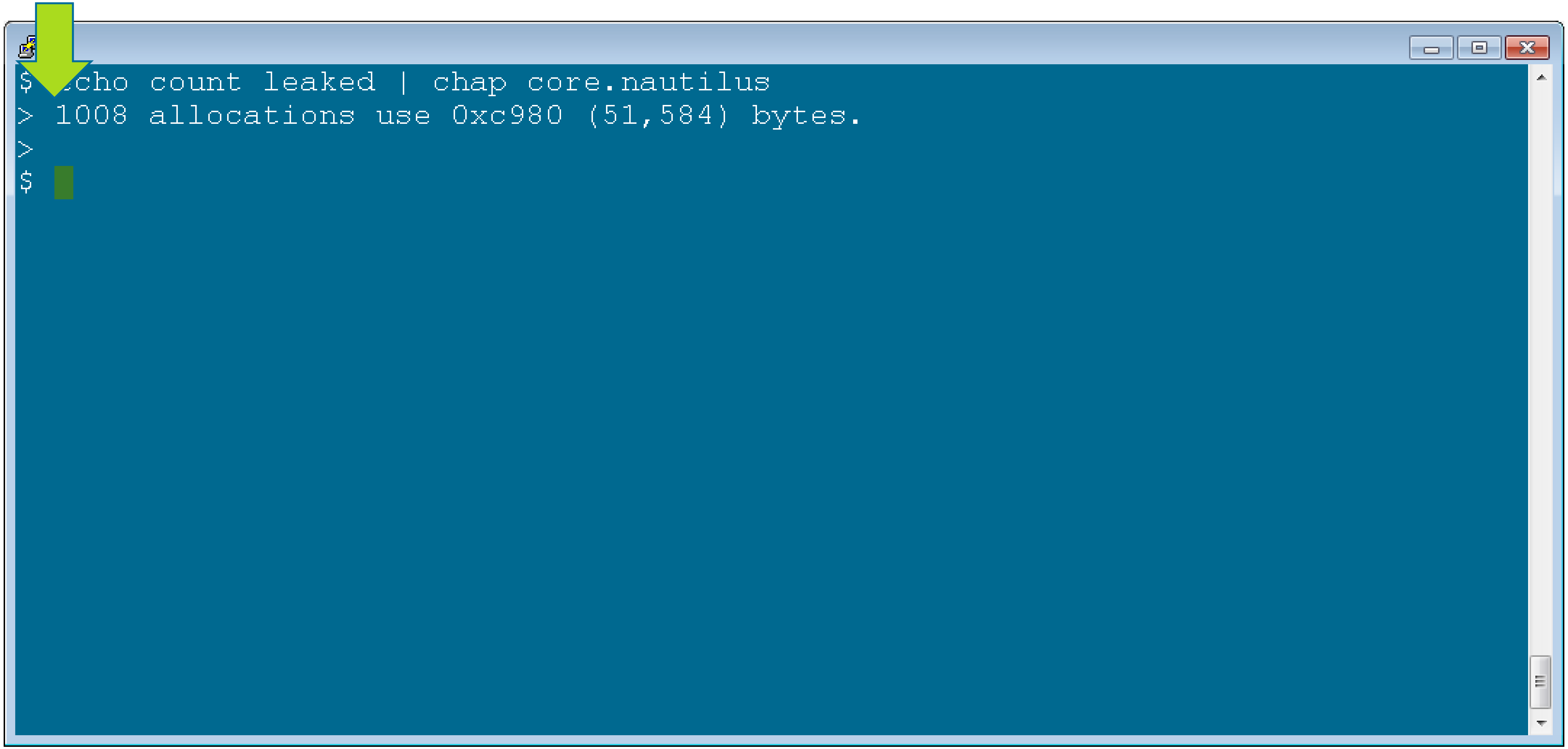
```
$ echo count leaked | chap core.nautilus
> 1008 allocations use 0xc980 (51,584) bytes.
>
$ 
```

The Simplest Use Case



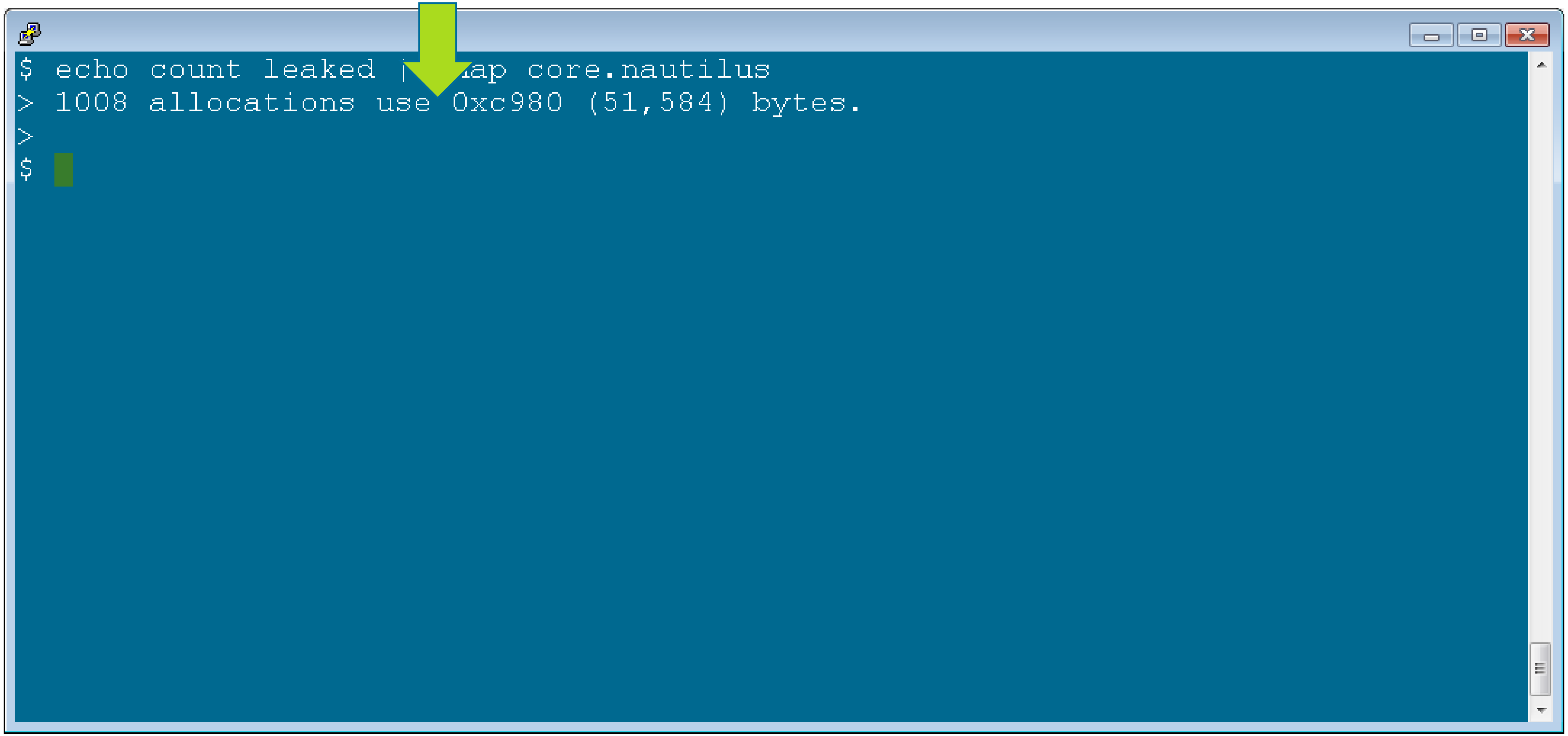
```
$ echo count leaked | chap core.nautilus
> 1008 allocations use 0xc980 (51,584) bytes.
>
$
```

The Simplest Use Case



```
$ echo count leaked | chap core.nautilus
> 1008 allocations use 0xc980 (51,584) bytes.
>
$ 
```

The Simplest Use Case



```
$ echo count leaked | chap core.nautilus
> 1008 allocations use 0xc980 (51,584) bytes.
>
$
```

Why Create Yet Another Memory Analysis Tool?

vmware®

Some Characteristics of Instrumentation Approaches

- Increase process size
- Have some performance penalty
- Distort timing
- Some alter allocation algorithms

Environments that Normally Run Without Instrumentation

- Customer production environments
- Performance tests
- Sizing tests
- Tests at scale
- Uptime tests

CHAP Finds Allocations

vmware®

Terminology: Allocations and Overhead

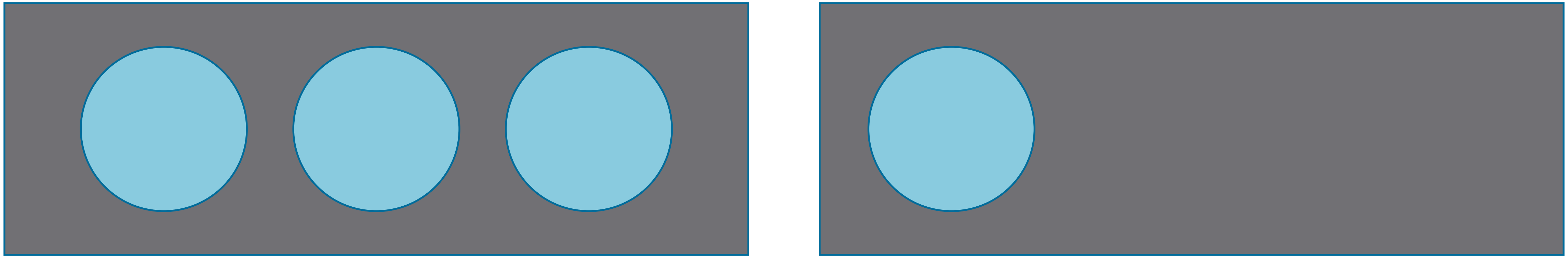
- A dynamic memory allocation function (e.g., malloc) provides a pointer to a sufficiently large **allocation**
- The **allocation** is considered **used** until it is returned to the allocator, when it becomes **free**
- Any writable memory used by the allocator beyond what is needed to hold every **used allocation** is considered **overhead.**
- Any writable memory other than **overhead** and **used allocations** is considered to be **outside of dynamic memory**

Terminology: Allocations and Overhead

- A dynamic memory allocation function (e.g., malloc) provides a pointer to a sufficiently large **allocation**
- The **allocation** is considered **used** until it is returned to the allocator, when it becomes **free**
- Any memory used by the allocator beyond what is needed to hold every **used allocation** is considered **overhead.**
- Any writable memory other than **overhead** and **used allocations** is considered to be **outside of dynamic memory**
- Allocations will be represented in this presentation by circles

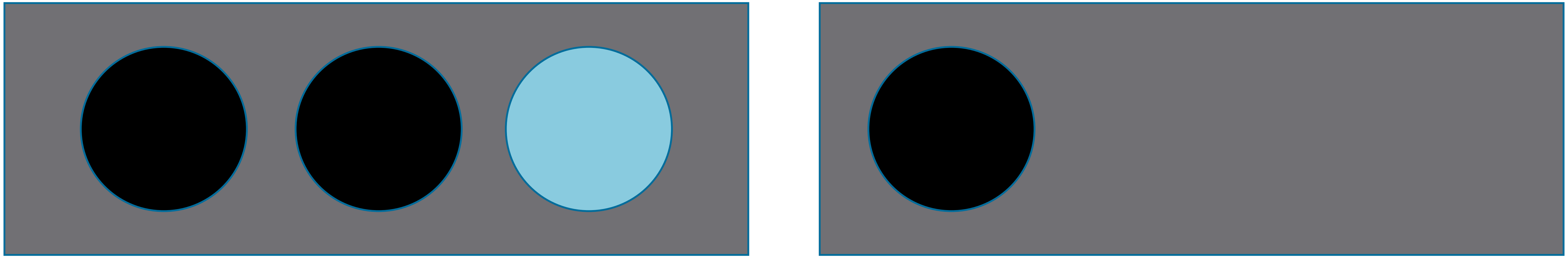
Some assumptions about allocators

- Satisfy requests for small **allocations** by partitioning larger ranges of memory
- Provide **allocations** that are “suitably aligned for any kind of variable”



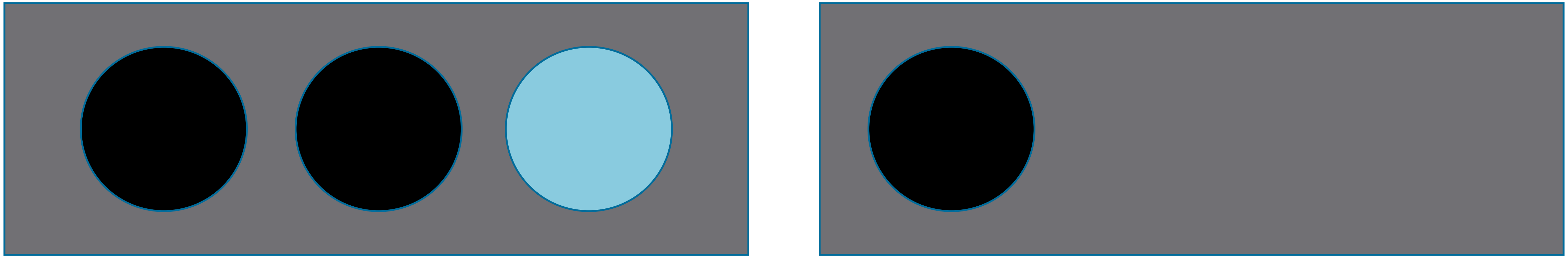
Some assumptions about allocators

- Satisfy requests for small **allocations** by partitioning larger ranges of memory
- Provide **allocations** that are “suitably aligned for any kind of variable”
- Allow **used allocations** to be freed



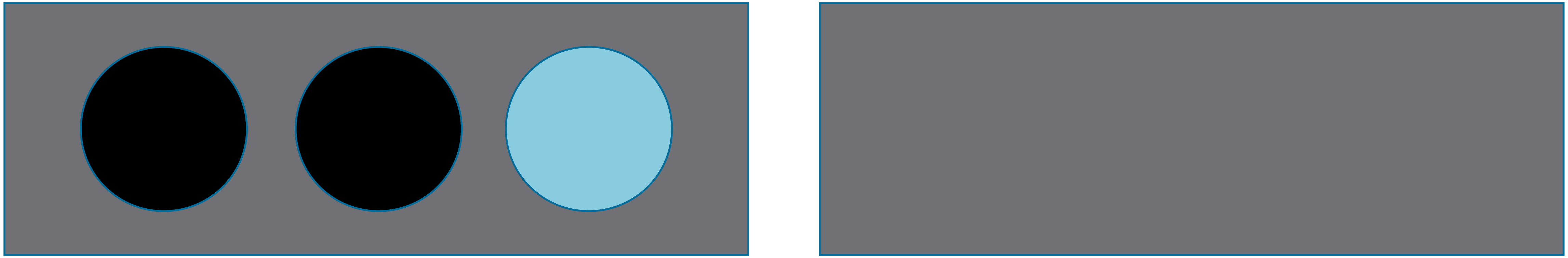
Some assumptions about allocators

- Satisfy requests for small **allocations** by partitioning larger ranges of memory
- Provide **allocations** that are “suitably aligned for any kind of variable”
- Allow **used allocations** to be freed
- Can free memory ranges that do not contain **used allocations**



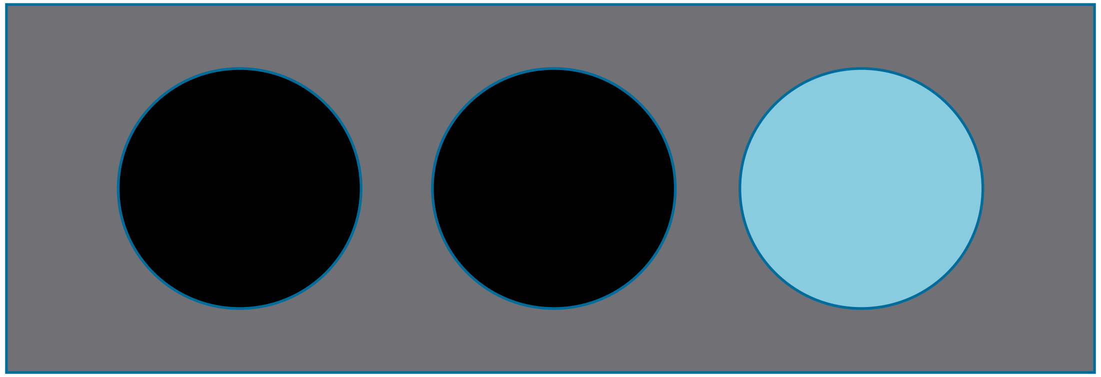
Some assumptions about allocators

- Satisfy requests for small **allocations** by partitioning larger ranges of memory
- Provide **allocations** that are “suitably aligned for any kind of variable”
- Allow **used allocations** to be freed
- Can free memory ranges that do not contain **used allocations**



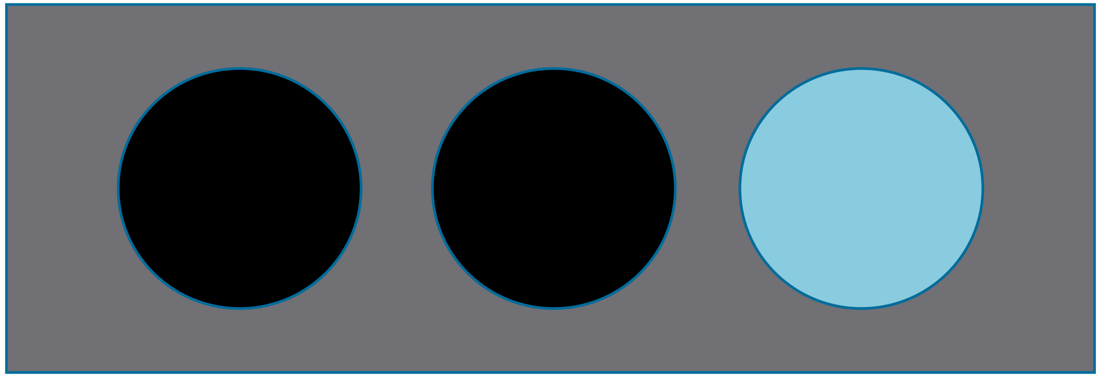
Some assumptions about allocators

- Satisfy requests for small **allocations** by partitioning larger ranges of memory
- Provide **allocations** that are “suitably aligned for any kind of variable”
- Allow **used allocations** to be freed
- Can free memory ranges that do not contain **used allocations**

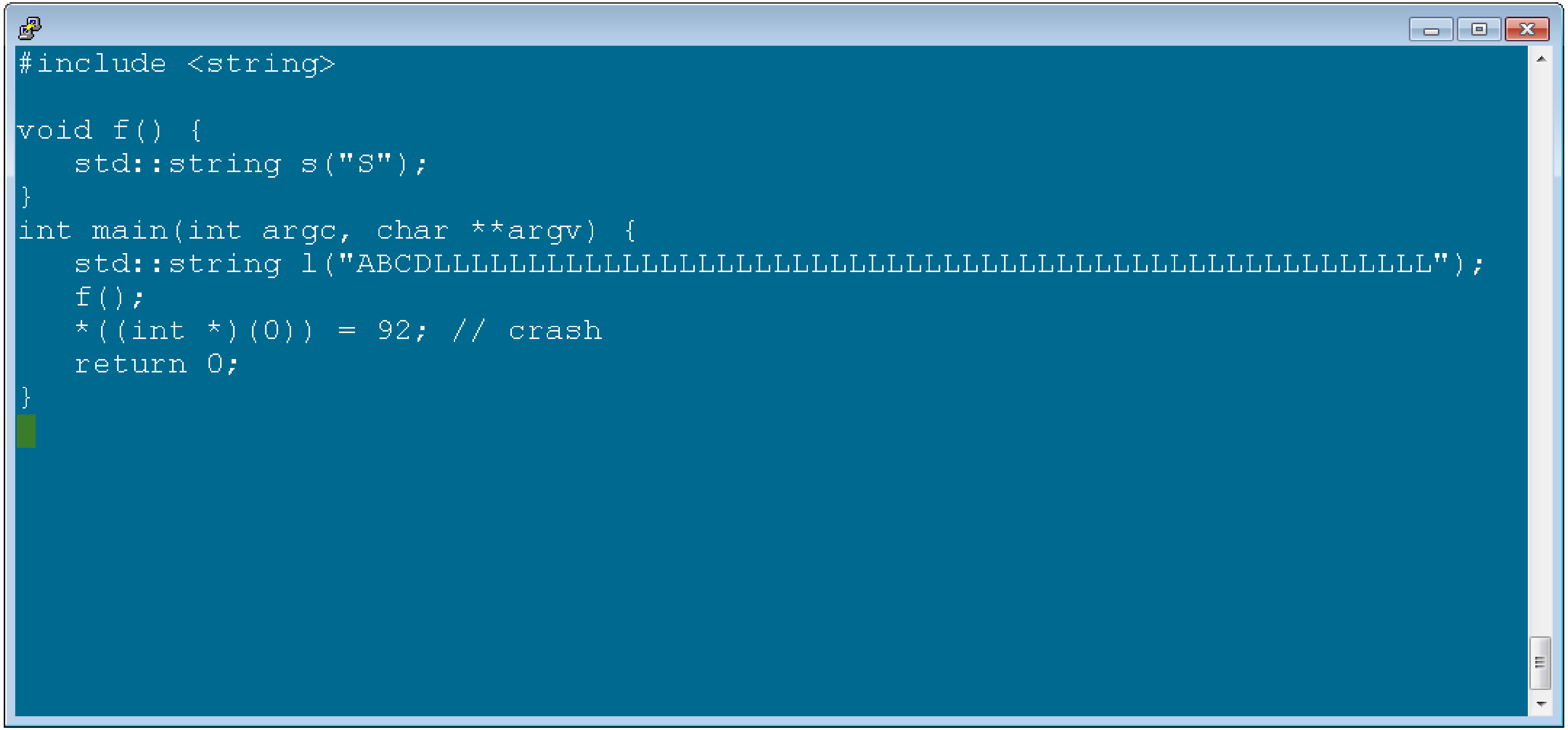


Some assumptions about allocators

- Satisfy requests for small **allocations** by partitioning larger ranges of memory
- Provide **allocations** that are “suitably aligned for any kind of variable”
- Allow **used allocations** to be freed
- Can free memory ranges that do not contain **used allocations**
- Often keep one or more **free allocation**, which can be used to satisfy some subsequent allocation request

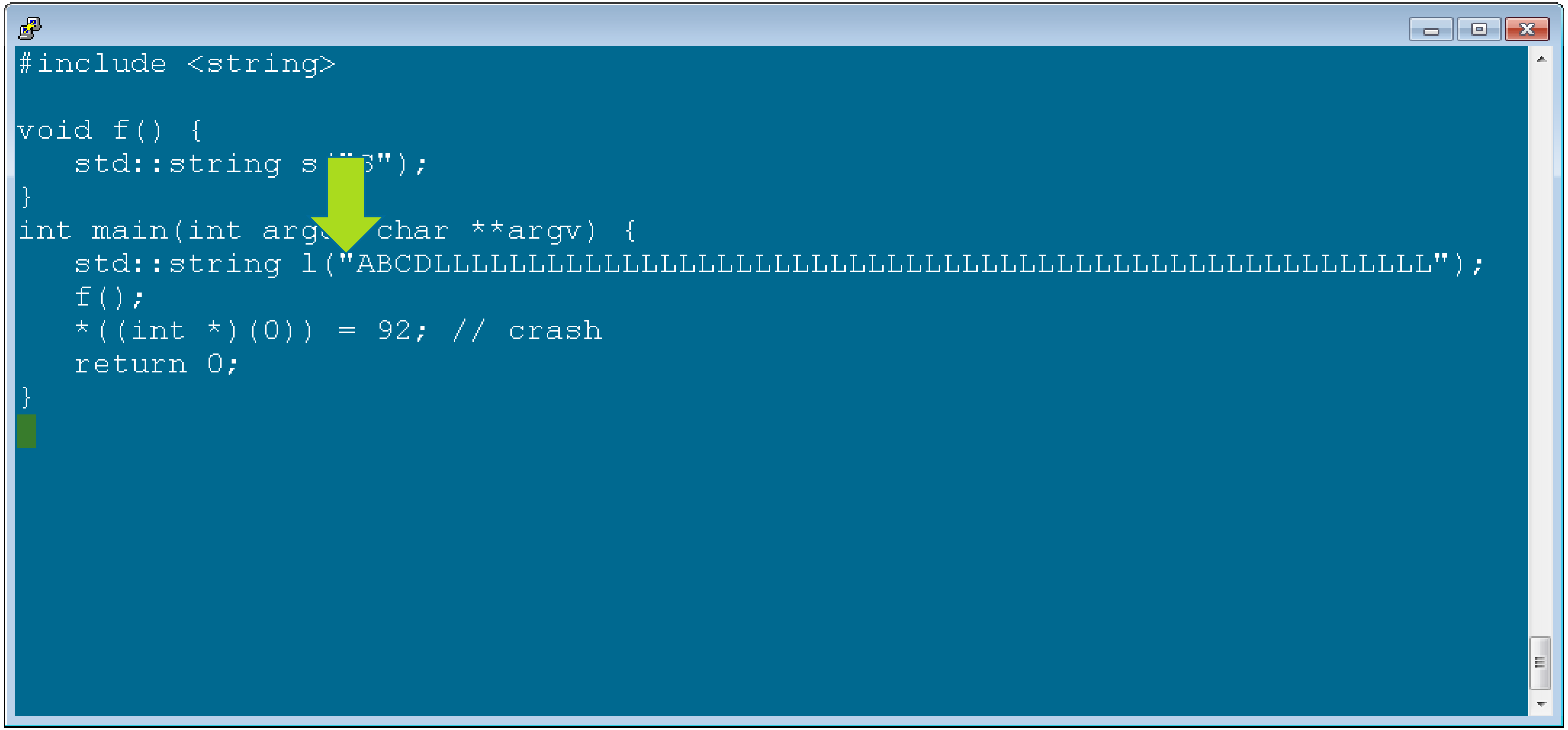


A Program To Illustrate Allocations



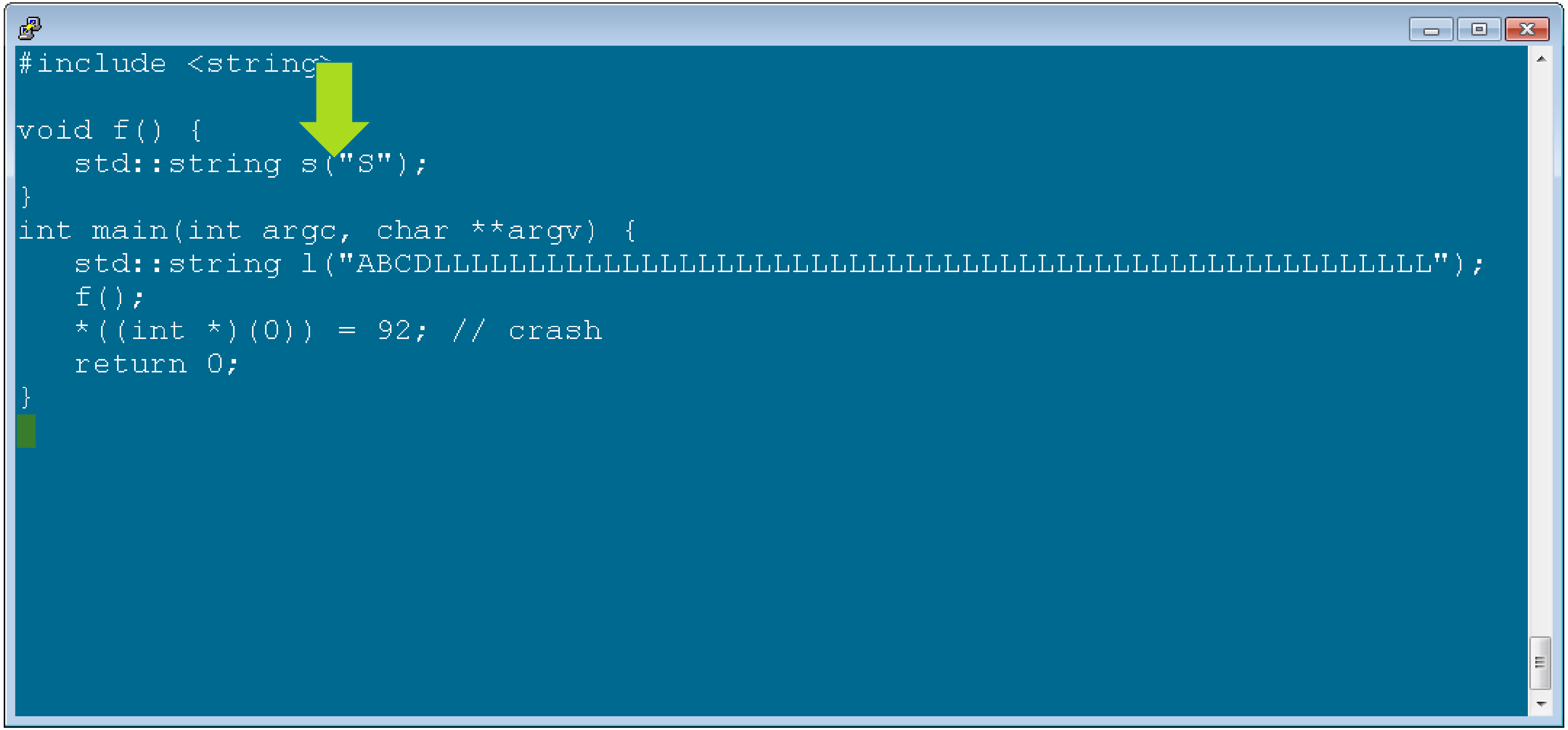
```
#include <string>

void f() {
   std::string s("S");
}
int main(int argc, char **argv) {
   std::string l("ABCDLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL");
   f();
   *((int *)(0)) = 92; // crash
   return 0;
}
```

A Program To Illustrate Allocations



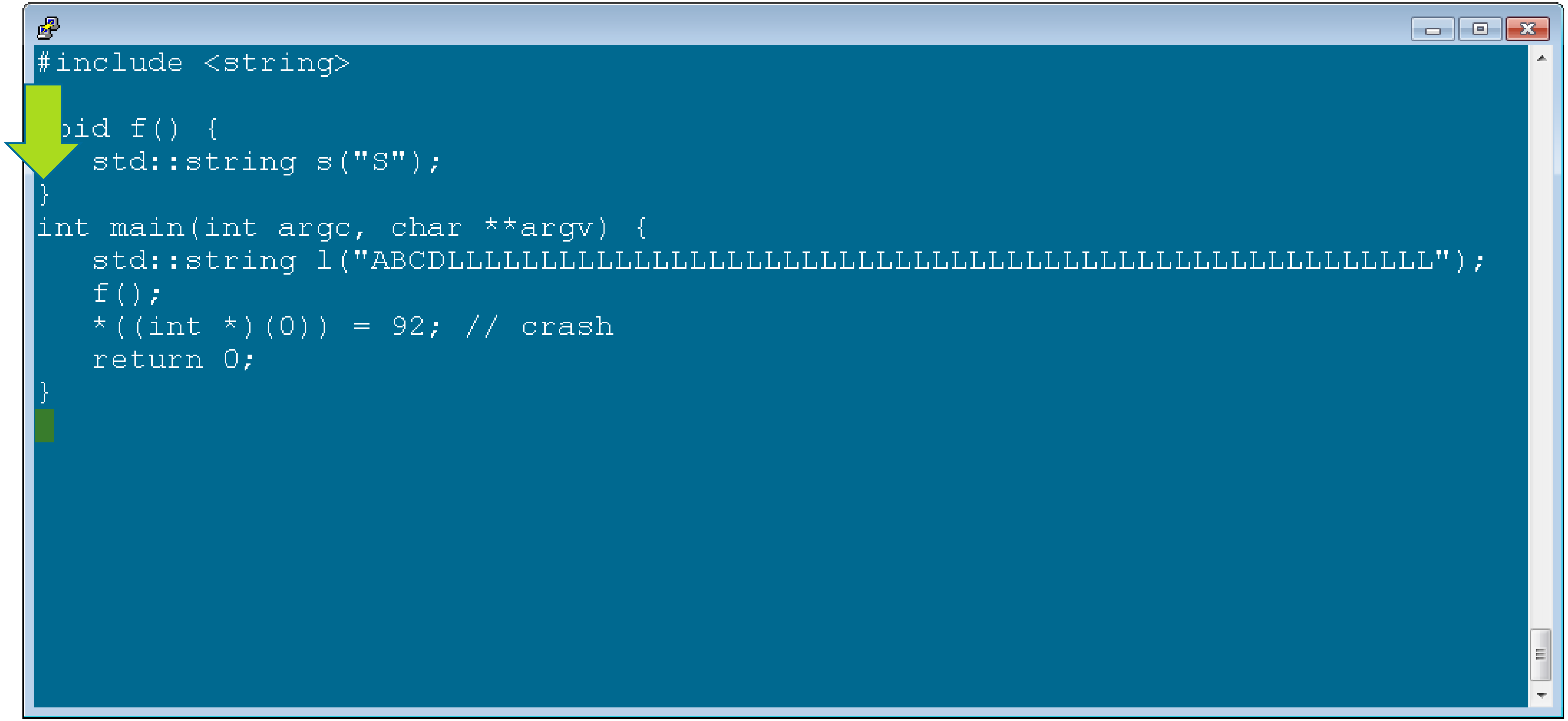
```
#include <string>

void f() {
   std::string s("B");
}
int main(int argc, char **argv) {
   std::string l("ABCDLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL");
   f();
   *((int *)(0)) = 92; // crash
   return 0;
}
```

A Program To Illustrate Allocations



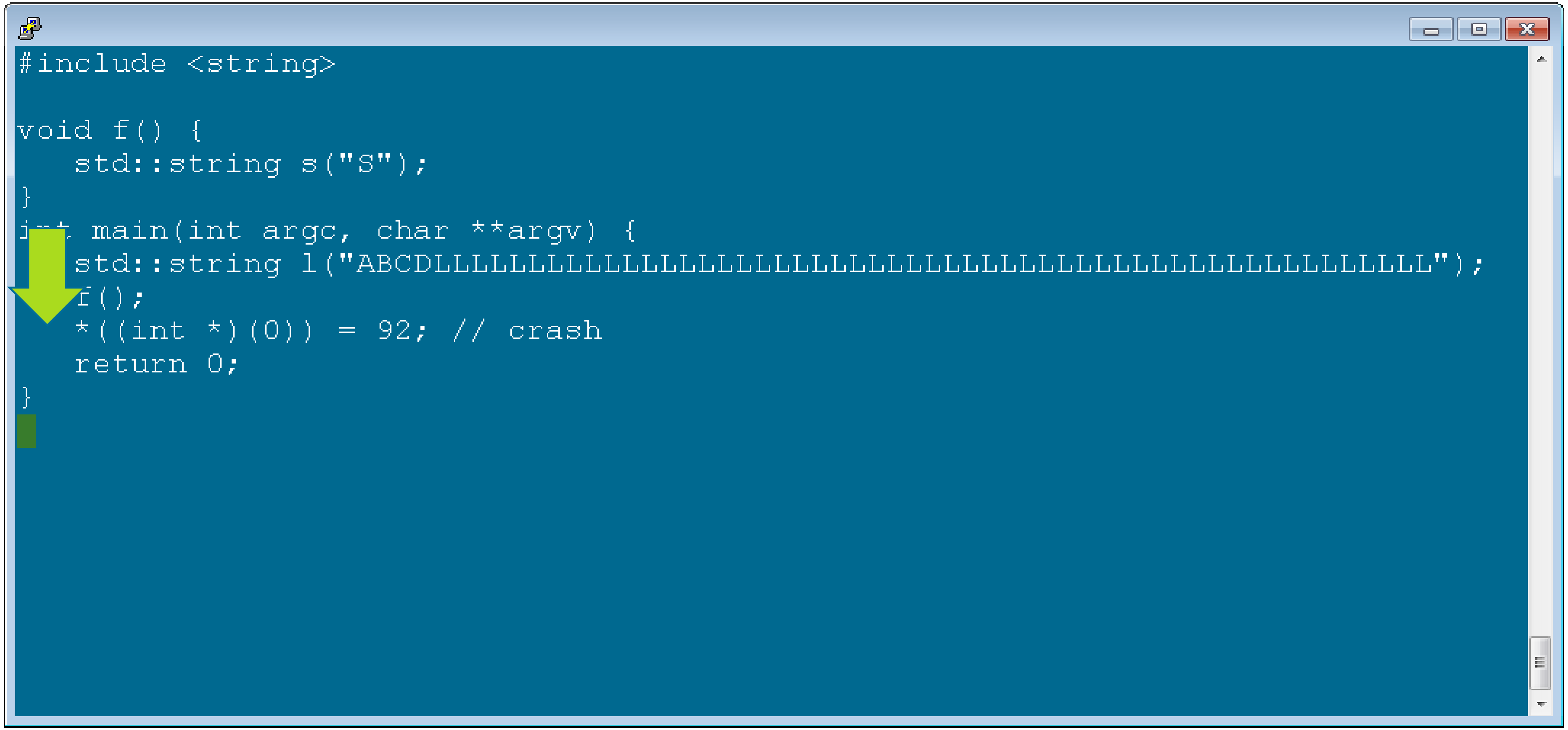
```
#include <string>

void f() {
   std::string s("S");
}
int main(int argc, char **argv) {
   std::string l("ABCDLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL");
   f();
   *((int *)(0)) = 92; // crash
   return 0;
}
```

A Program To Illustrate Allocations



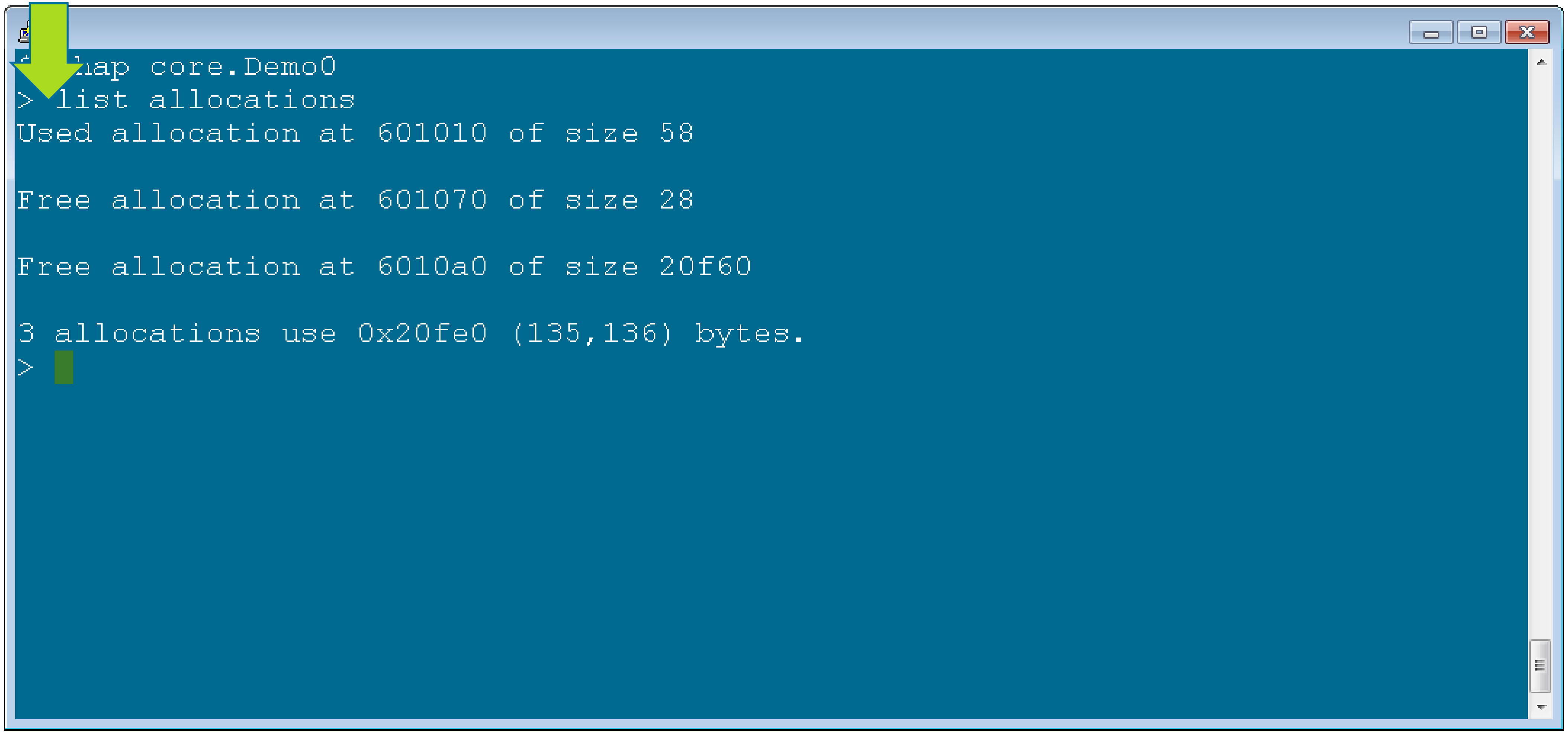
```
#include <string>

void f() {
   std::string s("S");
}
int main(int argc, char **argv) {
   std::string l("ABCDLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL");
   f();
   *((int *)(0)) = 92; // crash
   return 0;
}
```

A Program To Illustrate Allocations



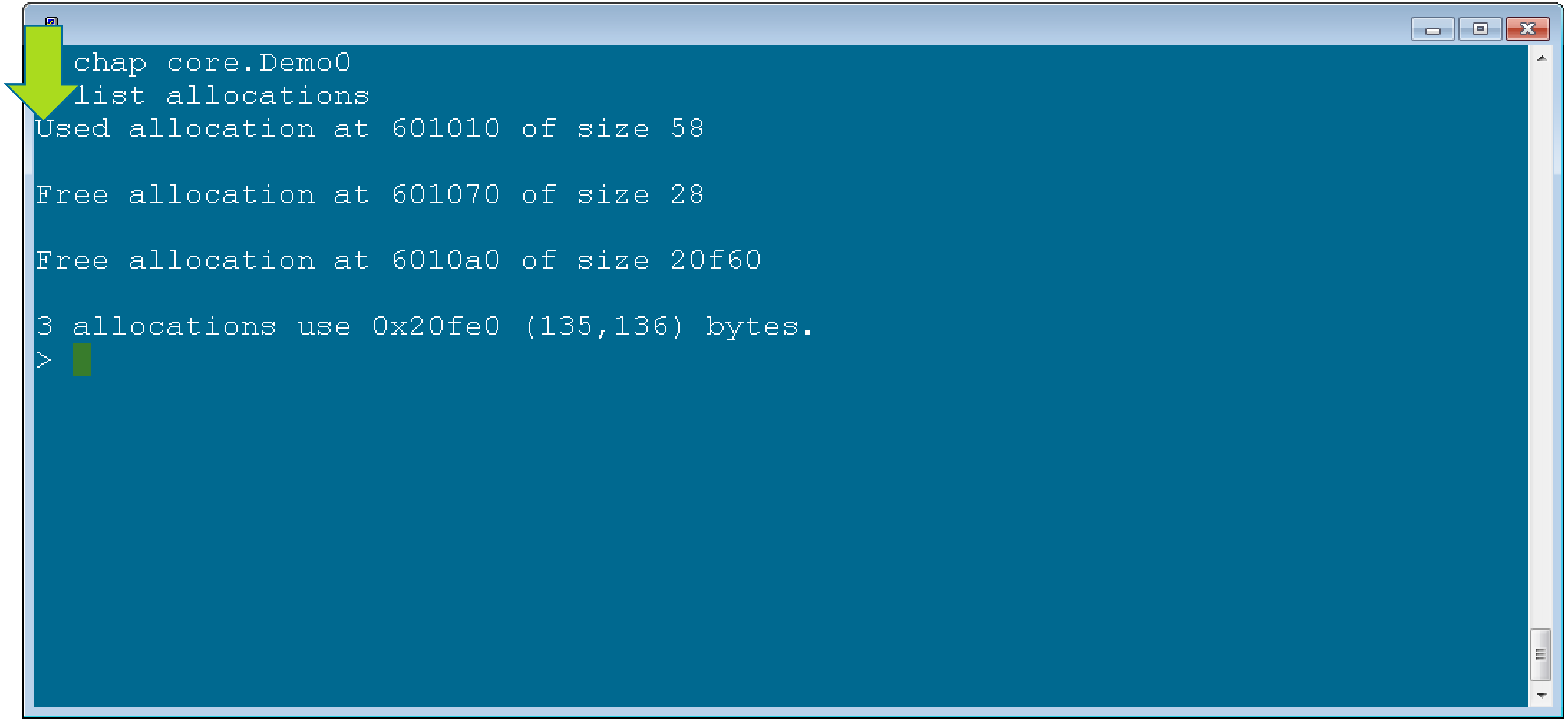
```
#include <string>

void f() {
   std::string s("S");
}
int main(int argc, char **argv) {
   std::string l("ABCDLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL");
   f();
   *((int *)(0)) = 92; // crash
   return 0;
}
```

Listing Allocations



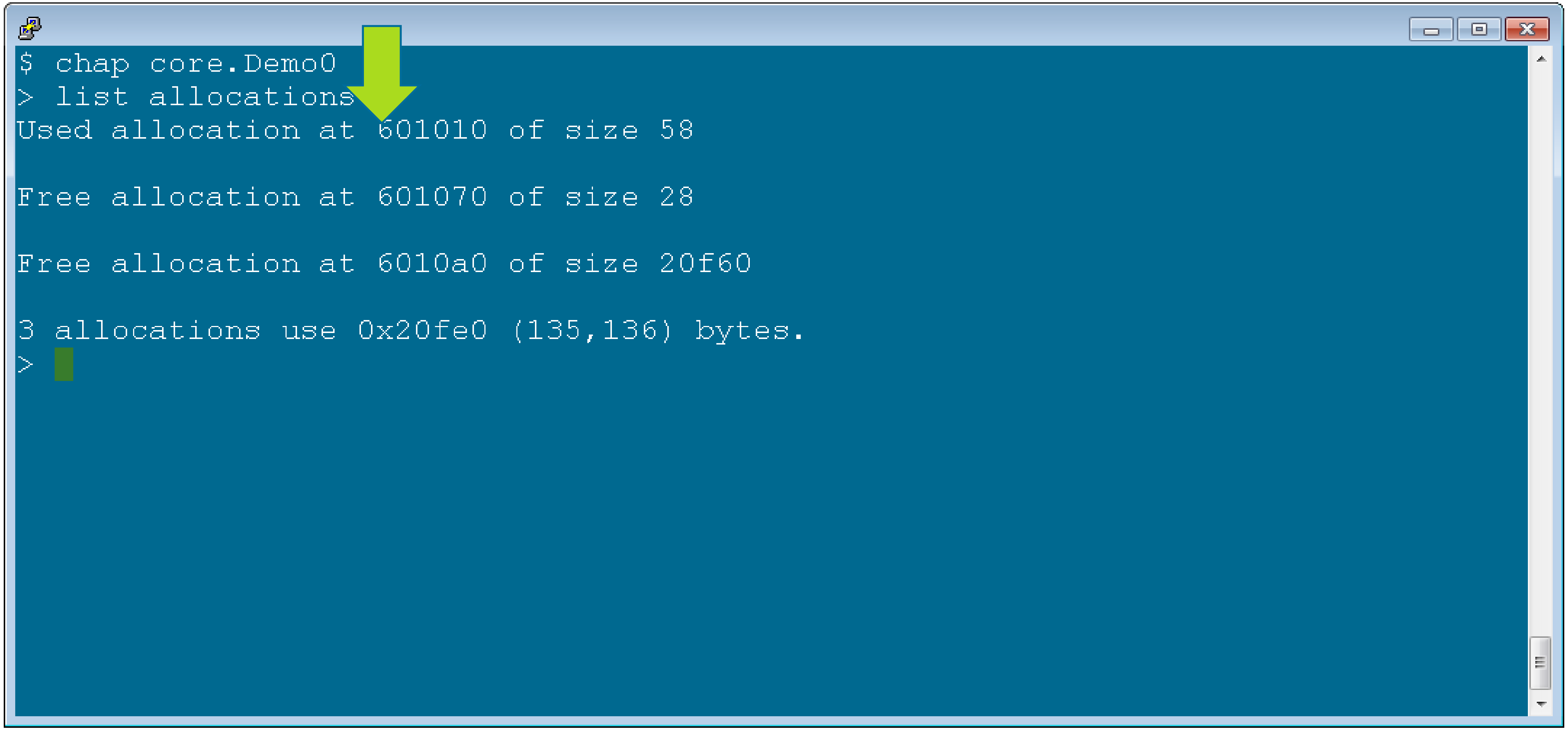
```
hap core.Demo0
> list allocations
Used allocation at 601010 of size 58

Free allocation at 601070 of size 28

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
>
```

Listing Allocations



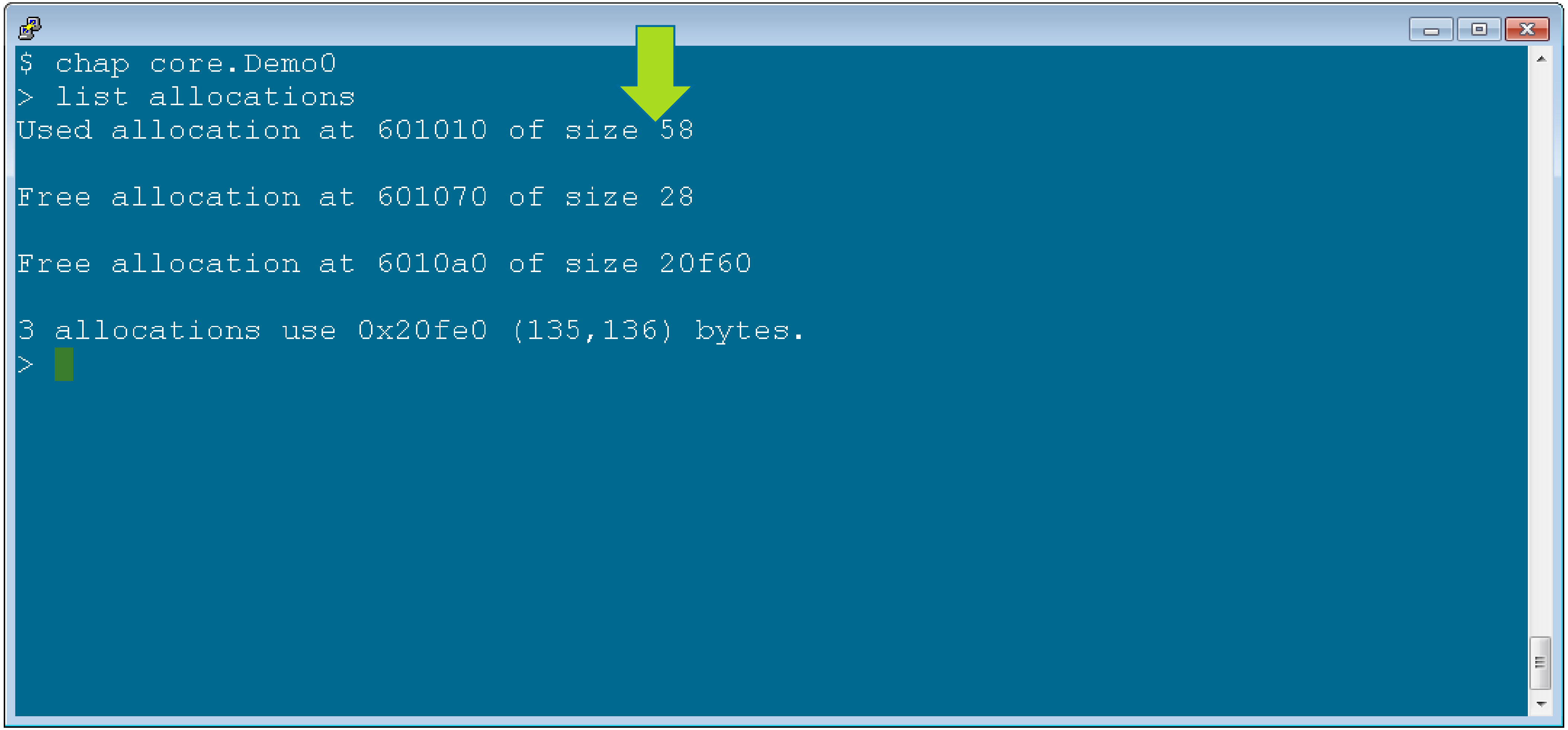
```
  chap core.Demo0
  list allocations
Used allocation at 601010 of size 58

Free allocation at 601070 of size 28

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
>
```

Listing Allocations



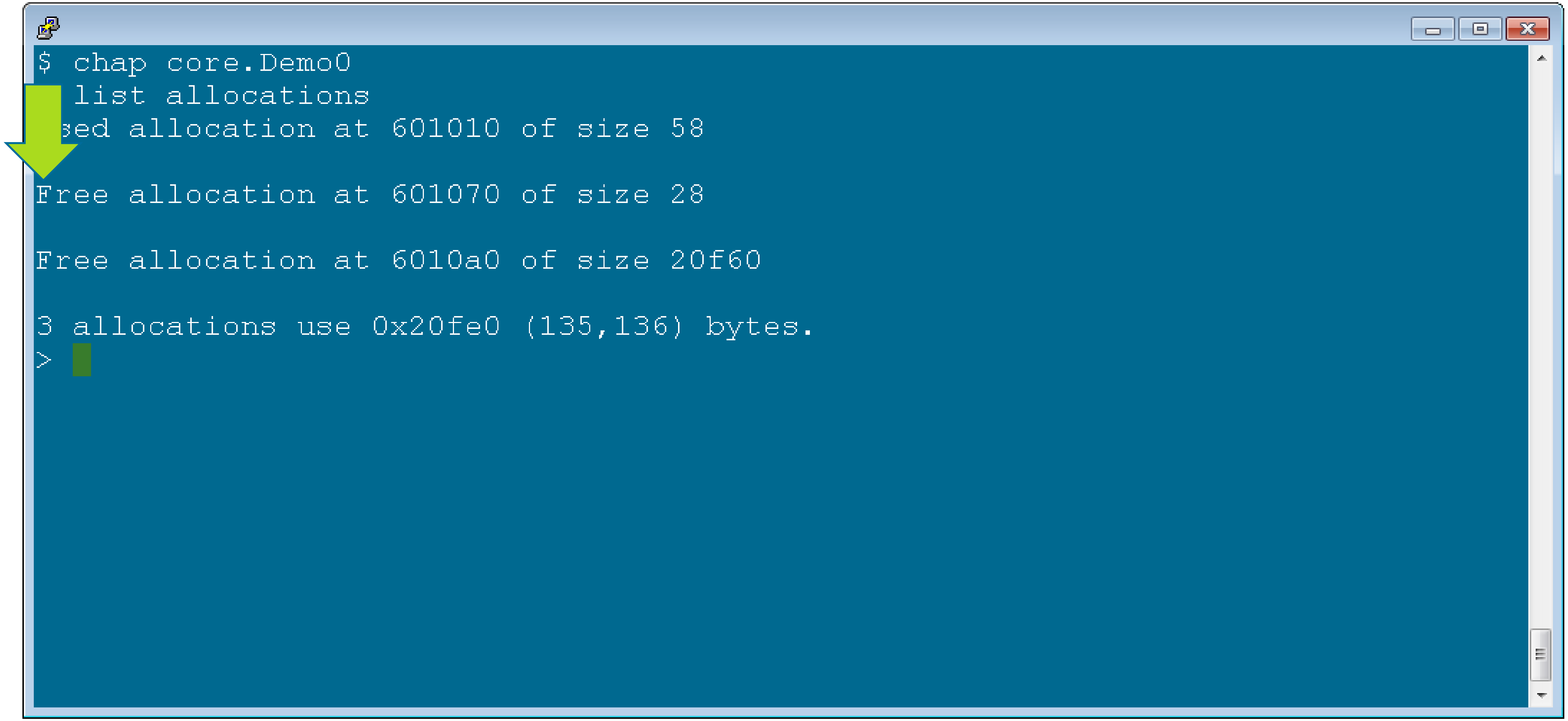
```
$ chap core.Demo0
> list allocations
Used allocation at 601010 of size 58

Free allocation at 601070 of size 28

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
> 
```

Listing Allocations



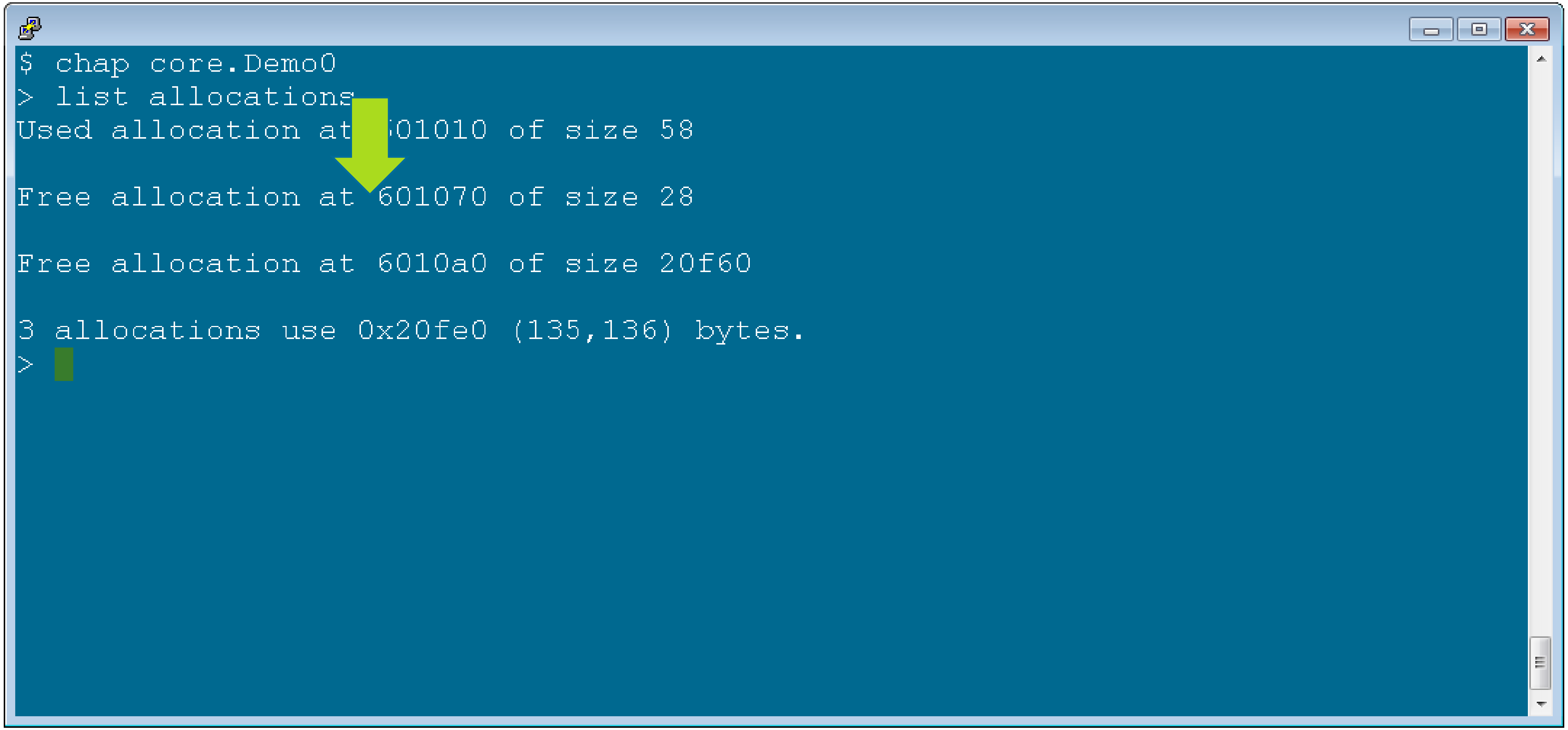
```
$ chap core.Demo0
> list allocations
Used allocation at 601010 of size 58

Free allocation at 601070 of size 28

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
>
```

Listing Allocations



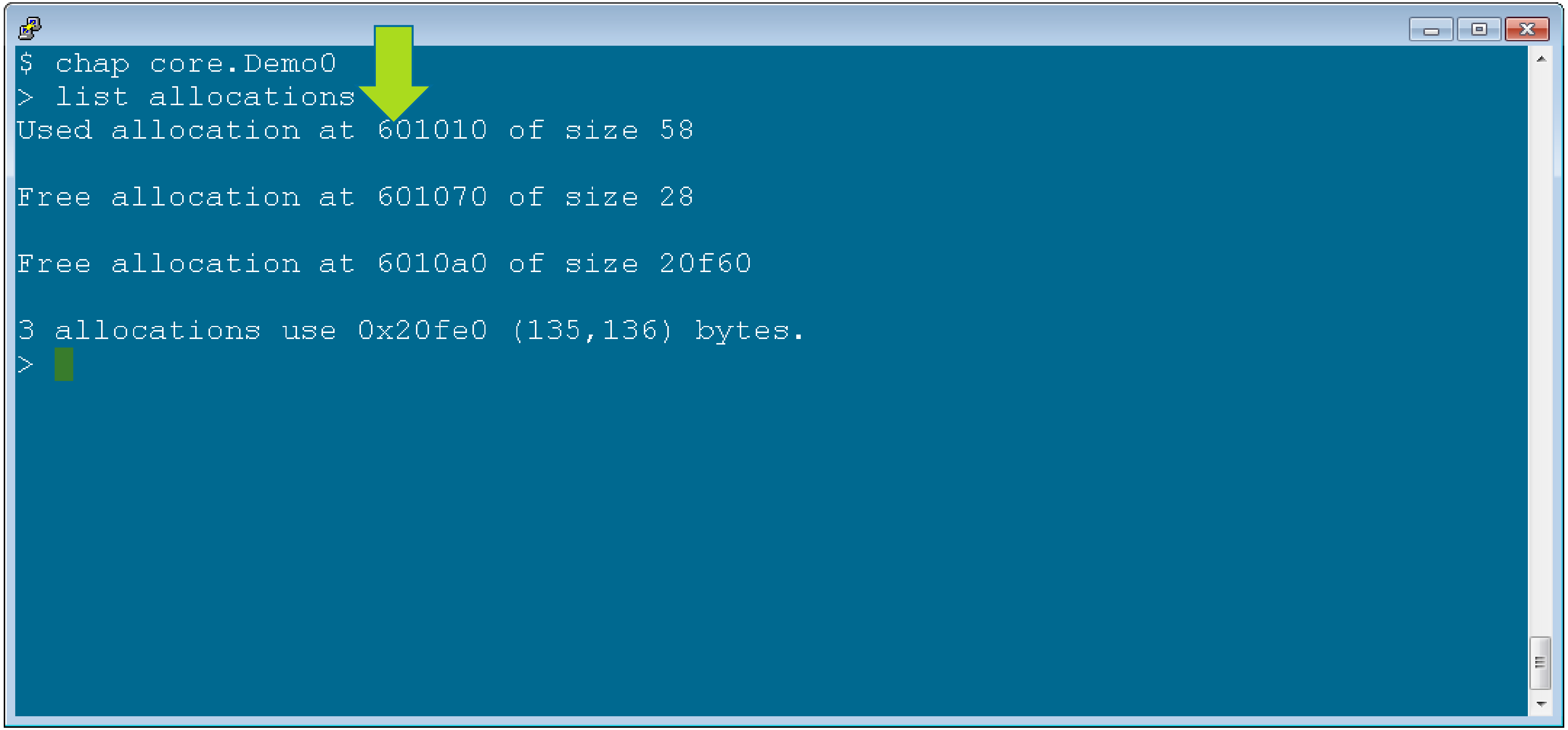
```
$ chap core.Demo0
  list allocations
 sed allocation at 601010 of size 58

Free allocation at 601070 of size 28

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
>
```

Listing Allocations



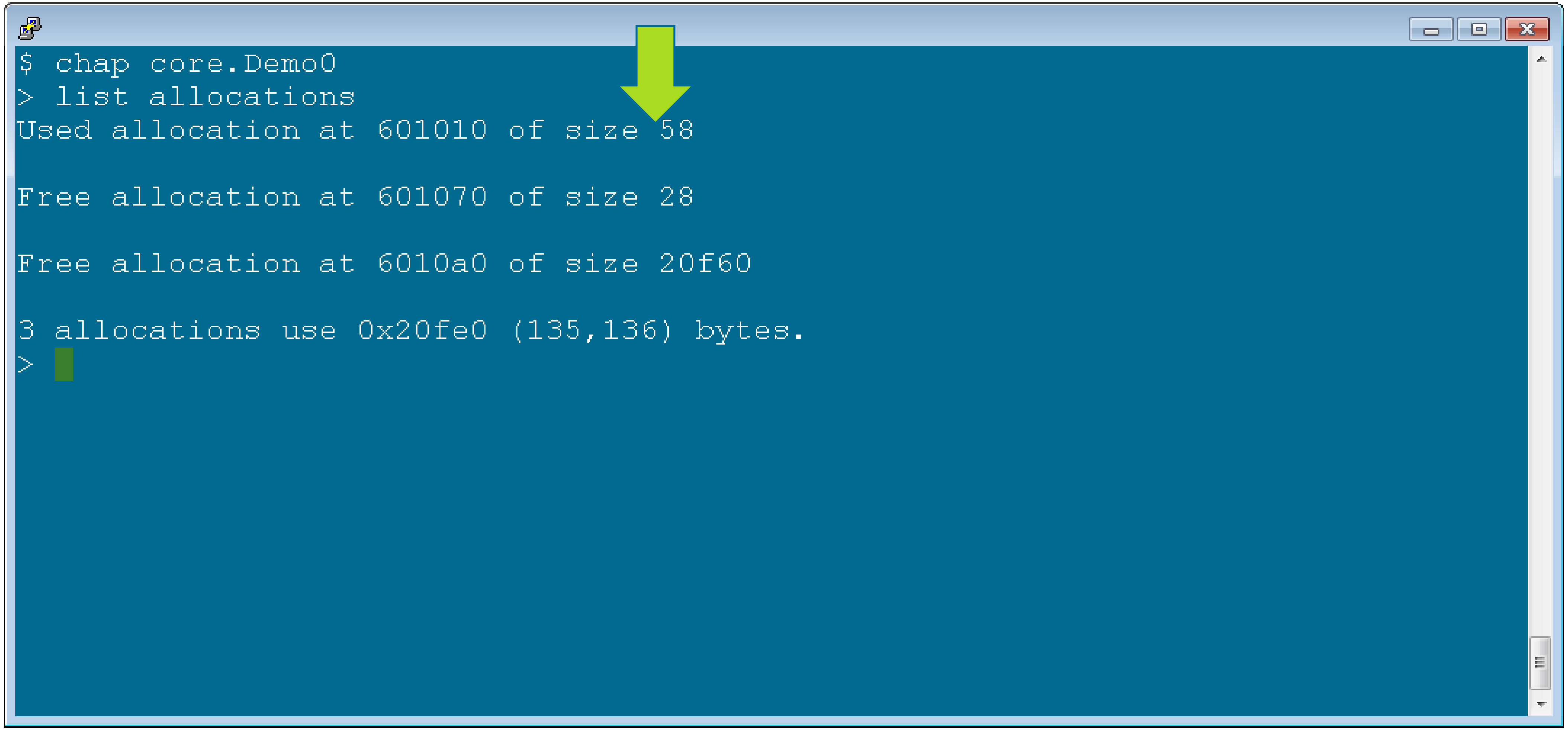
```
$ chap core.Demo0
> list allocations
Used allocation at 601010 of size 58

Free allocation at 601070 of size 28

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
> 
```

Listing Allocations



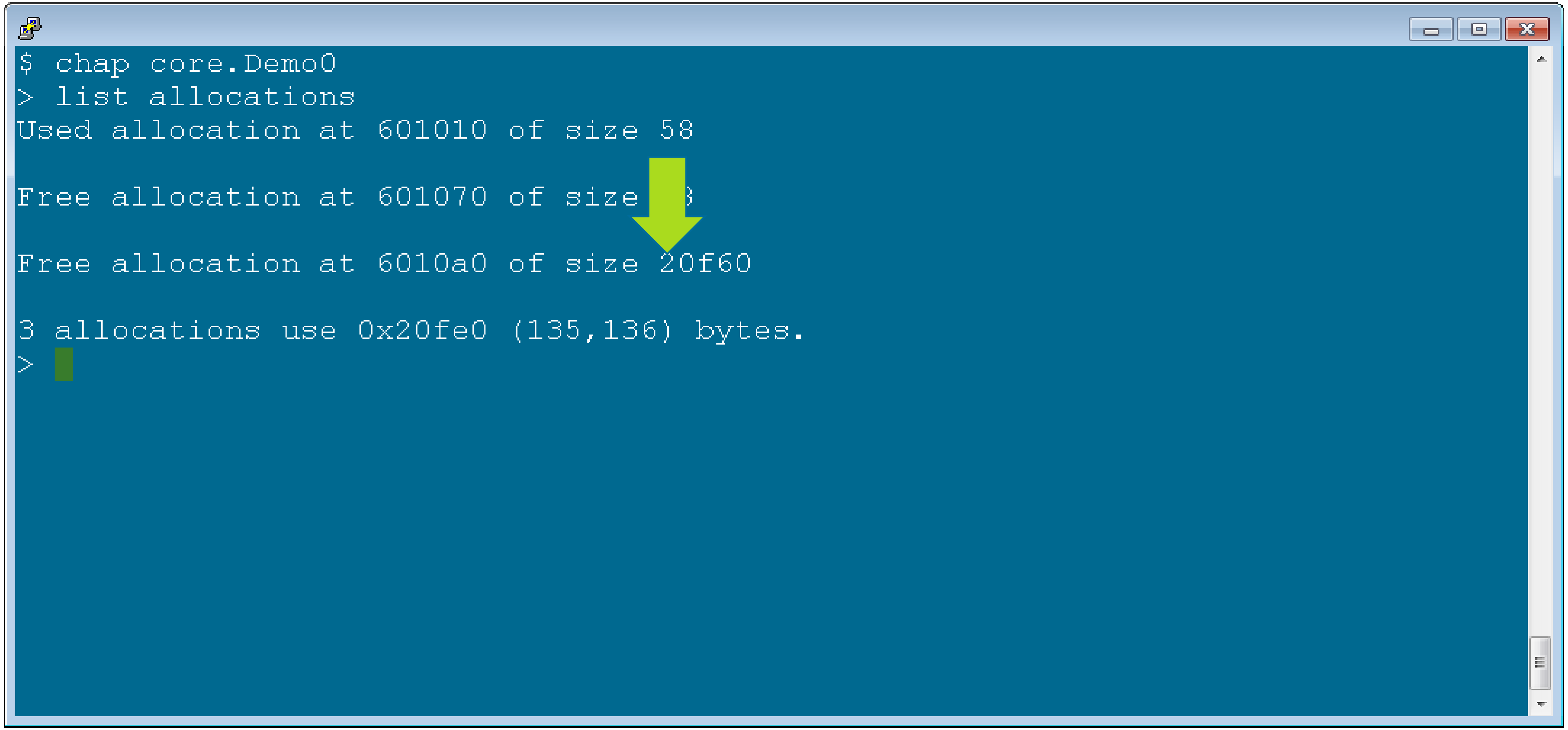
```
$ chap core.Demo0
> list allocations
Used allocation at 601010 of size 58

Free allocation at 601070 of size 28

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
>
```

Listing Allocations



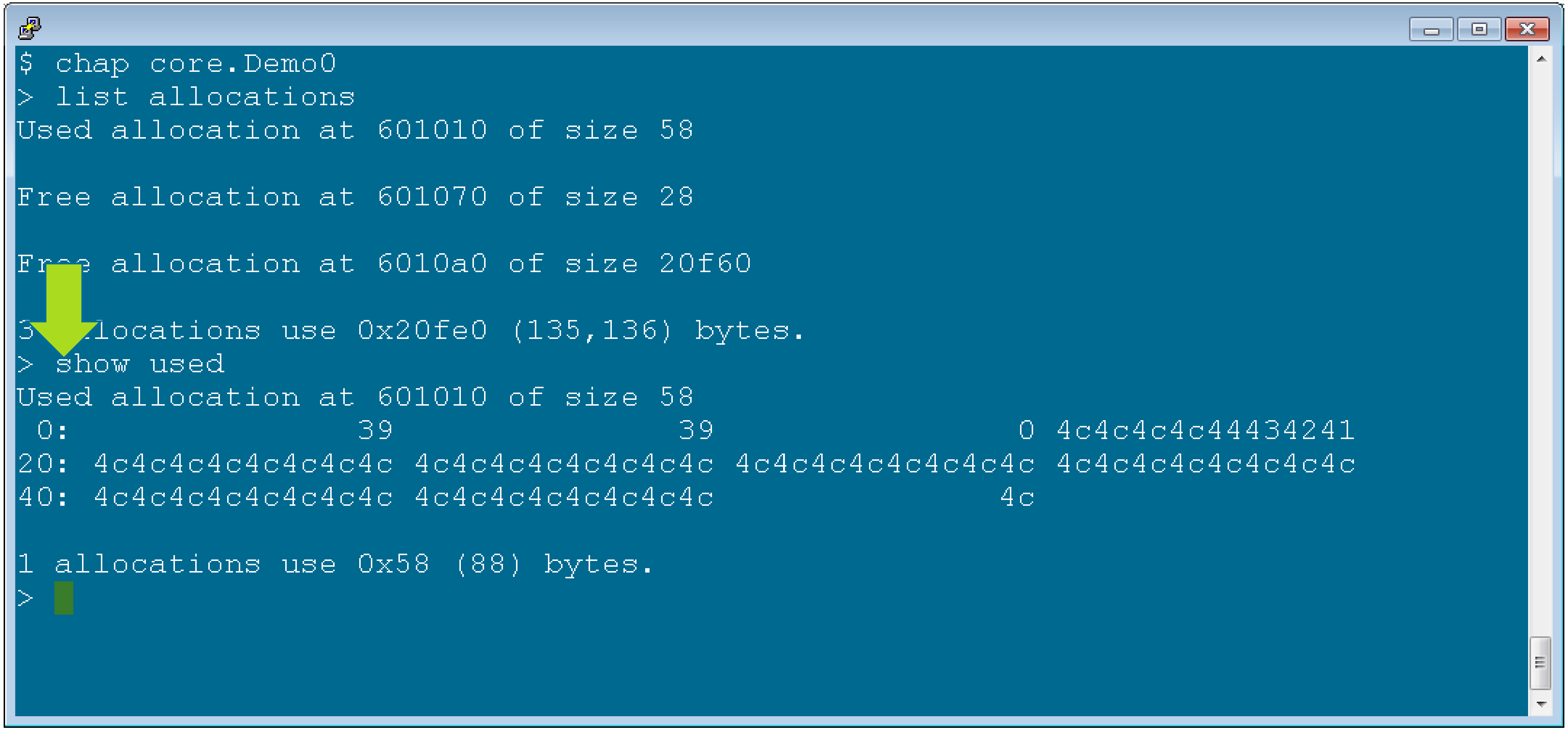
```
$ chap core.Demo0
> list allocations
Used allocation at 601010 of size 58

Free allocation at 601070 of size 28

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
> 
```

Listing Allocations



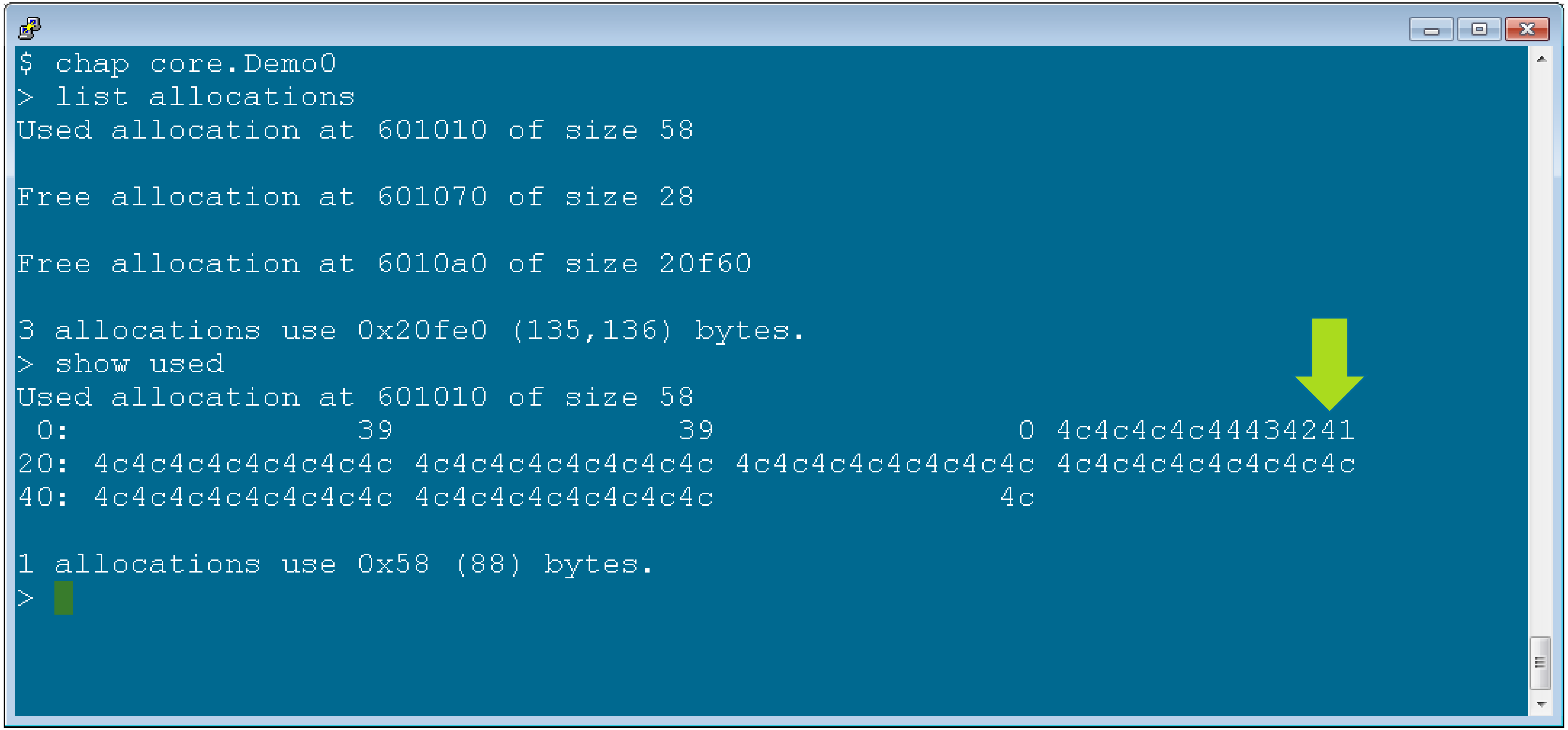
```
$ chap core.Demo0
> list allocations
Used allocation at 601010 of size 58

Free allocation at 601070 of size

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
>
```

Showing Used Allocations



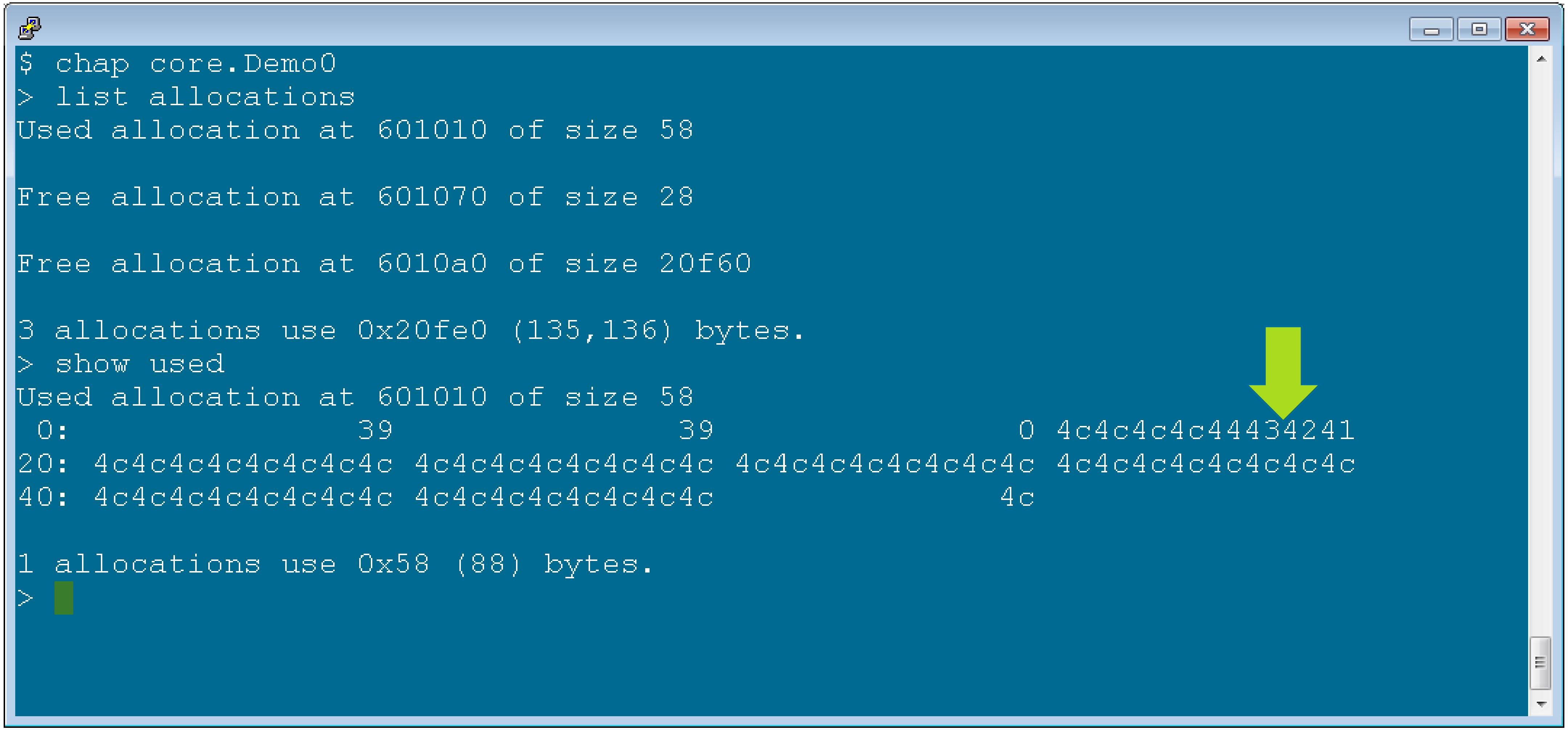
```
$ chap core.Demo0
> list allocations
Used allocation at 601010 of size 58

Free allocation at 601070 of size 28

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
> show used
Used allocation at 601010 of size 58
 0:               39               39                0 4c4c4c4c44434241
20: 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c
40: 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c               4c

1 allocations use 0x58 (88) bytes.
>
```

Showing Used Allocations



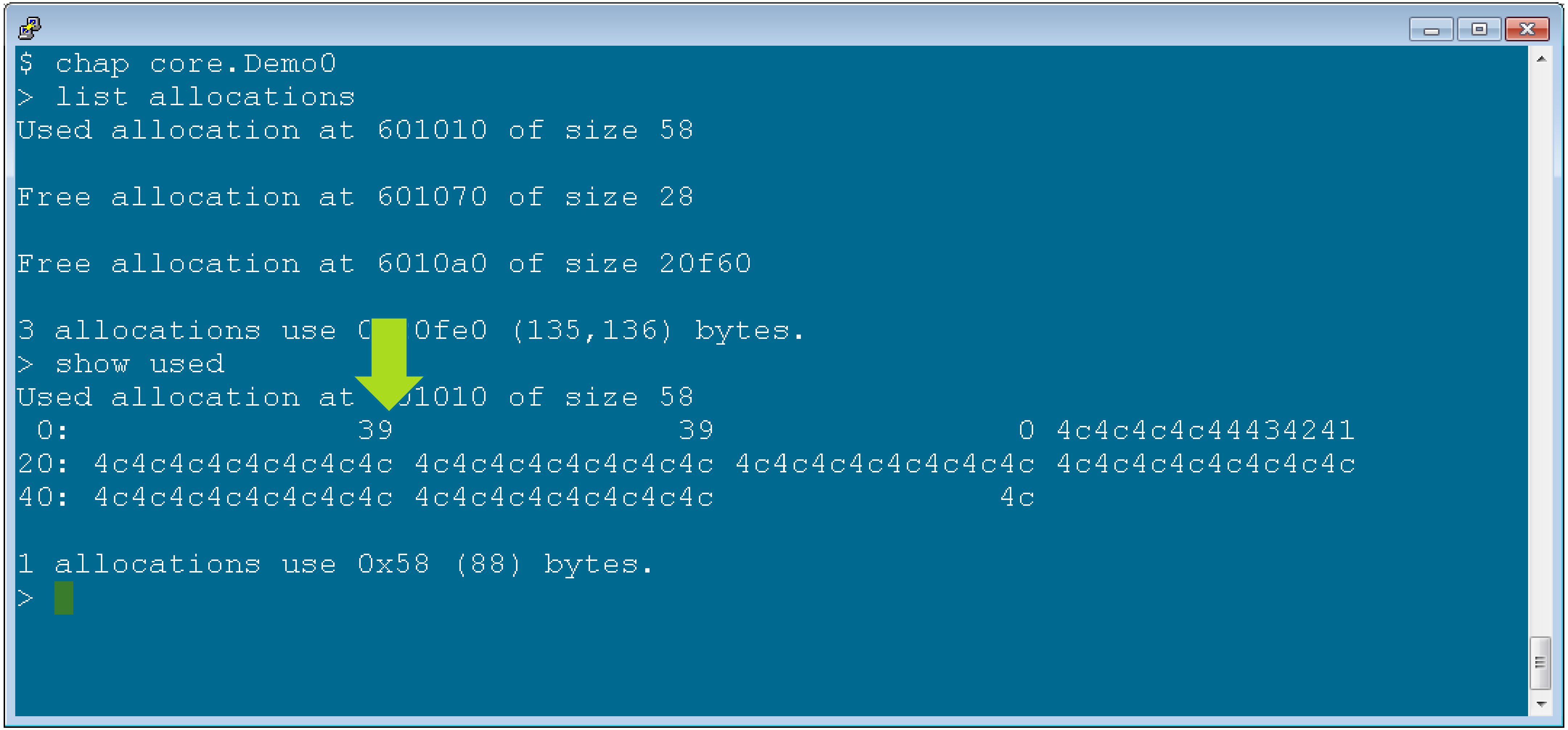
```
$ chap core.Demo0
> list allocations
Used allocation at 601010 of size 58

Free allocation at 601070 of size 28

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
> show used
Used allocation at 601010 of size 58
 0:               39               39                0 4c4c4c4c44434241
20: 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c
40: 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c               4c

1 allocations use 0x58 (88) bytes.
>
```

Showing Used Allocations



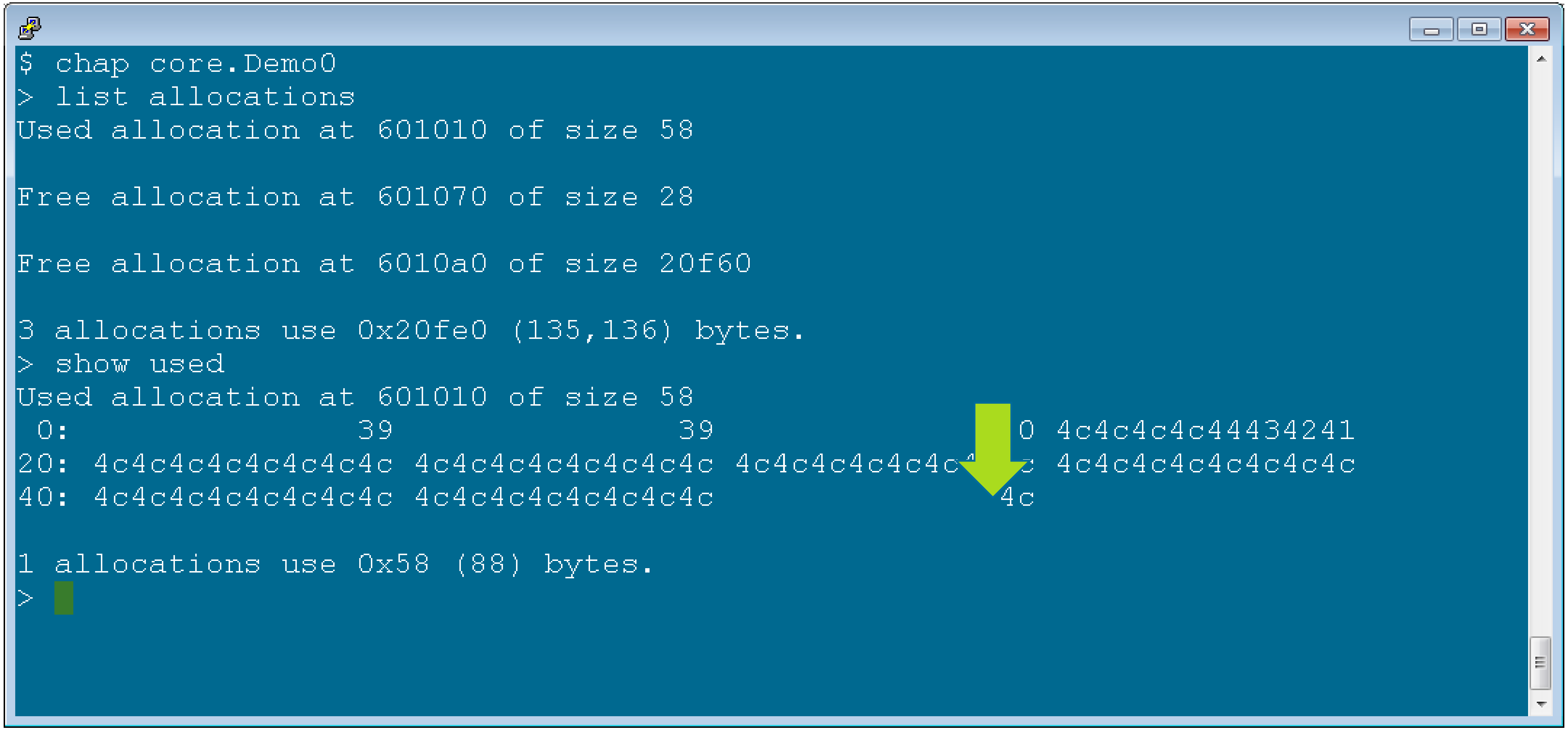
```
$ chap core.Demo0
> list allocations
Used allocation at 601010 of size 58

Free allocation at 601070 of size 28

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
> show used
Used allocation at 601010 of size 58
 0:               39               39                0 4c4c4c4c44434241
20: 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c
40: 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c               4c

1 allocations use 0x58 (88) bytes.
> 
```

Showing Used Allocations



```
$ chap core.Demo0
> list allocations
Used allocation at 601010 of size 58

Free allocation at 601070 of size 28

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
> show used
Used allocation at 601010 of size 58
 0:                39                39                 0 4c4c4c4c44434241
20: 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c
40: 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c                4c

1 allocations use 0x58 (88) bytes.
>
```

Showing Used Allocations

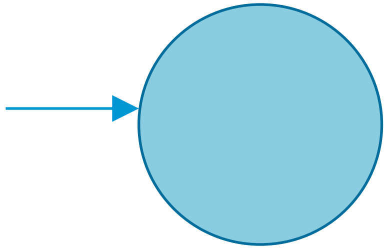


```
$ chap core.Demo0
> list allocations
Used allocation at 601010 of size 58

Free allocation at 601070 of size 28

Free allocation at 6010a0 of size 20f60

3 allocations use 0x20fe0 (135,136) bytes.
> show used
Used allocation at 601010 of size 58
 0:               39               39                0 4c4c4c4c44434241
20: 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c
40: 4c4c4c4c4c4c4c4c 4c4c4c4c4c4c4c4c               4c

1 allocations use 0x58 (88) bytes.
> 
```

Chap Finds References to Allocations

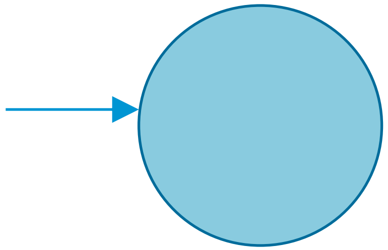
Terminology: Reference

- A **reference** to an **allocation** is a value somewhere (possibly in a register or in memory) paired with some interpretation of that value as providing a live pointer to some part of the **allocation**
- A **real reference** to an **allocation** is a **reference** tor which the interpretation is correct
- A **false reference** to an **allocation** is a **reference** tor which the interpretation is incorrect
- A **missed reference** to an **allocation** is a **reference** that is not detected

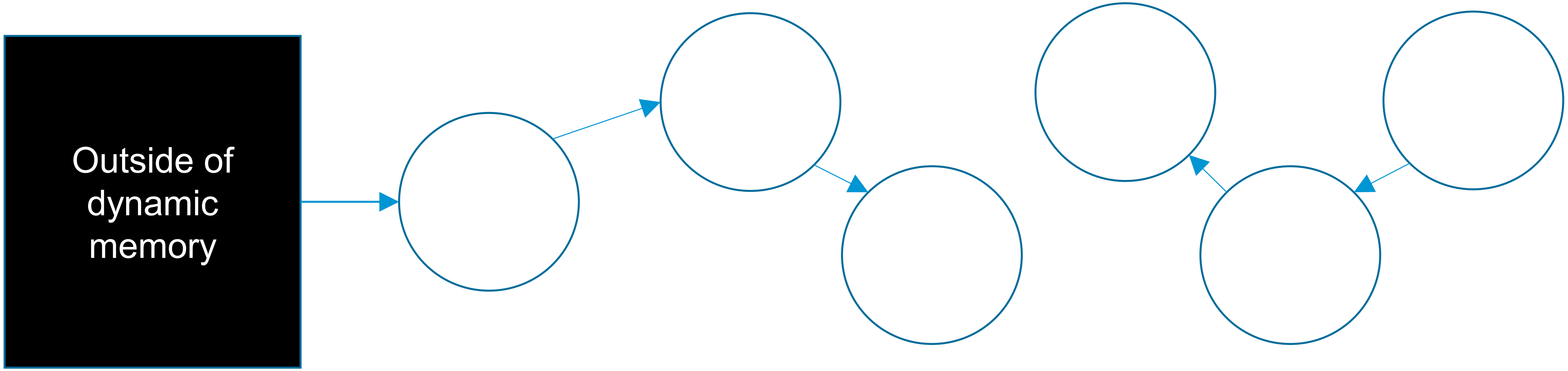


Examples of References

- A register associated with some thread contains a live pointer p to some part of an allocation
- A pointer-sized range of memory contains a live pointer p to some part of an allocation
- A register or memory contains f(p), e.g. myEncryptionFunction(p)
- Somewhere entirely outside the process holds p or f(p)

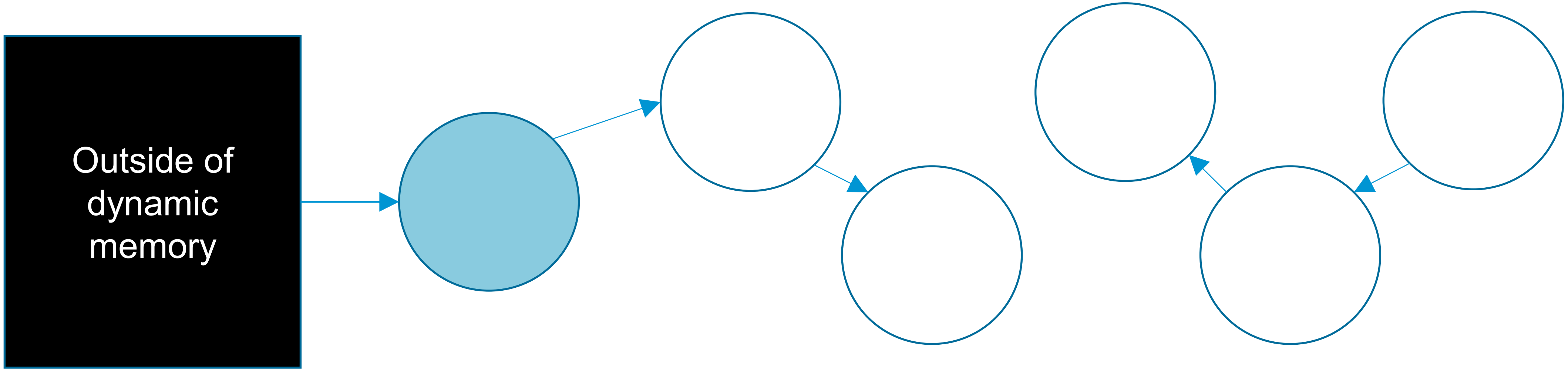


References and Allocations Form a Directed Graph



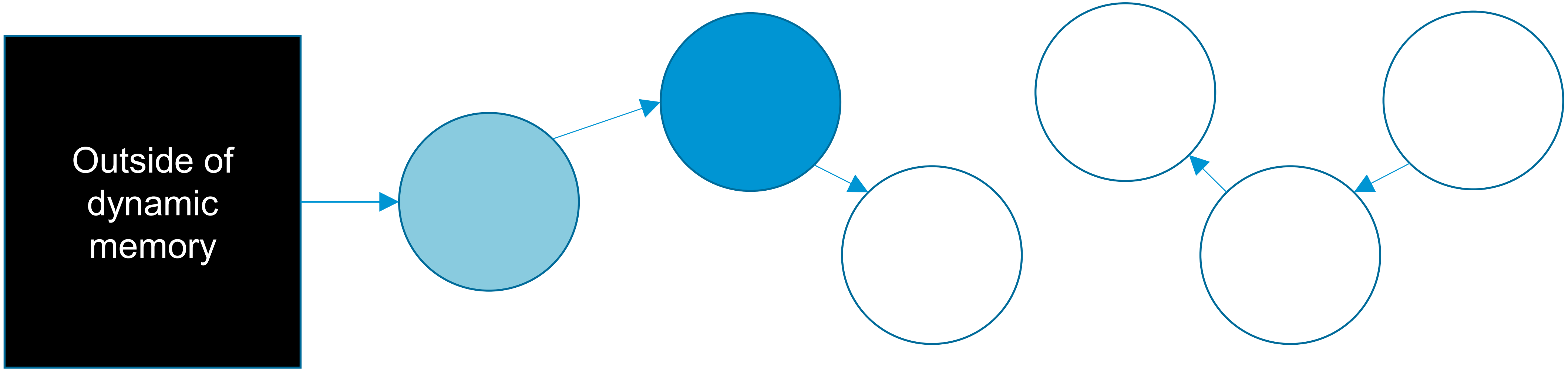
Terminology: Anchored and Leaked Allocations

- A **used allocation** is considered an **anchor point** if it is directly referenced from **outside of dynamic memory**



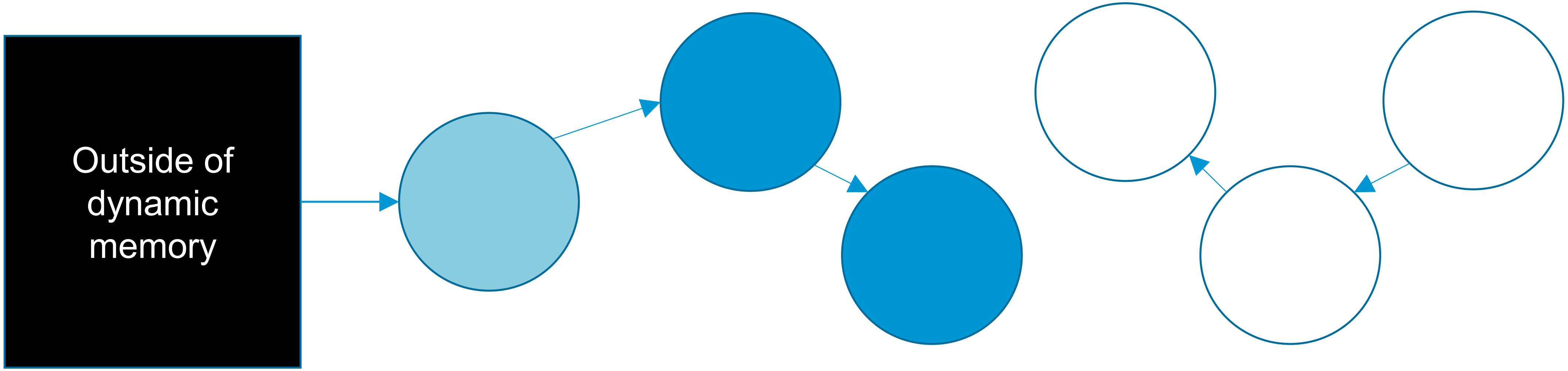
Terminology: Anchored and Leaked Allocations

- A **used allocation** is considered an **anchor point** if it is directly referenced from **outside of dynamic memory**
- A **used allocation** is considered to be **anchored** if it is an **anchor point** or is referenced by an **anchored allocation**



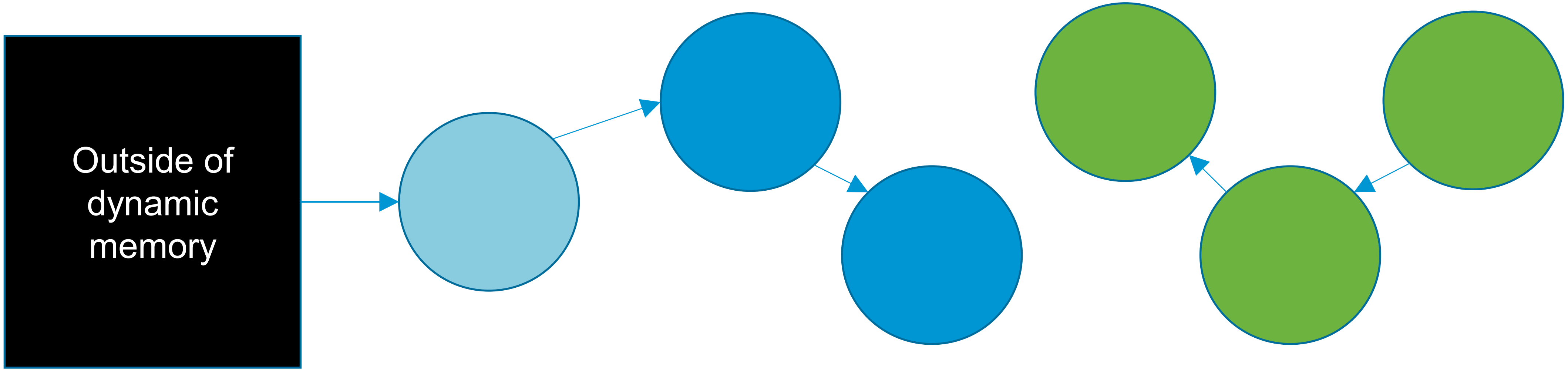
Terminology: Anchored and Leaked Allocations

- A **used allocation** is considered an **anchor point** if it is directly referenced from **outside of dynamic memory**
- A **used allocation** is considered to be **anchored** if it is an **anchor point** or is referenced by an **anchored allocation**



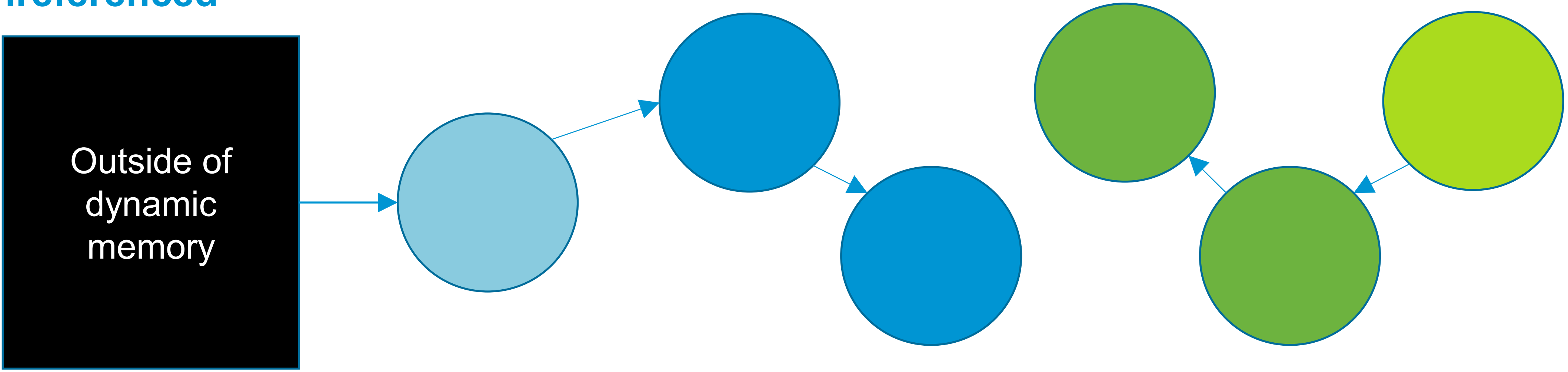
Terminology: Anchored and Leaked Allocations

- A **used allocation** is considered an **anchor point** if it is directly referenced from **outside of dynamic memory**
- A **used allocation** is considered to be **anchored** if it is an **anchor point** or is referenced by an **anchored allocation**
- A **used allocation** that is not **anchored** is considered to be **leaked**

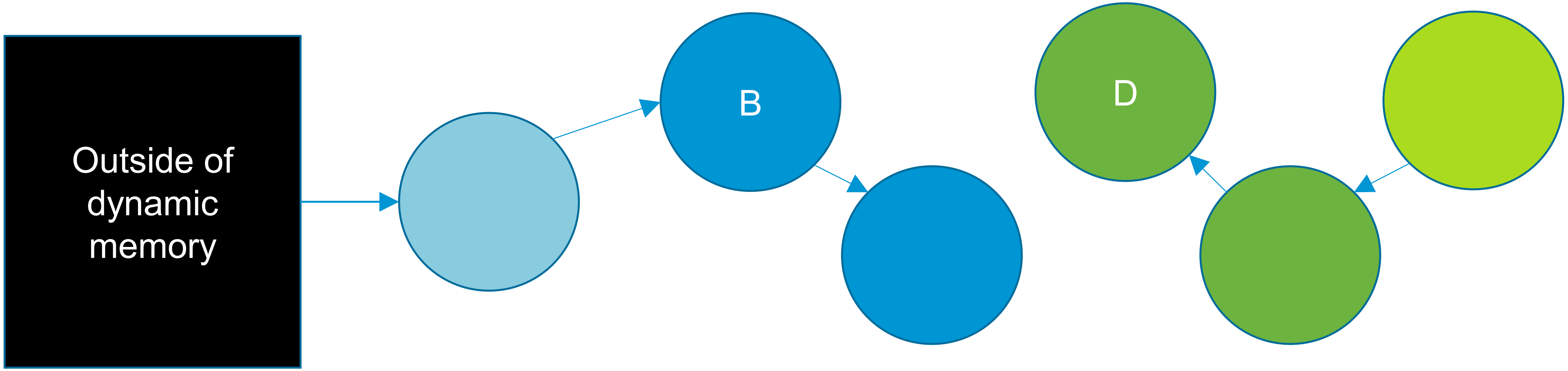


Terminology: Anchored and Leaked Allocations

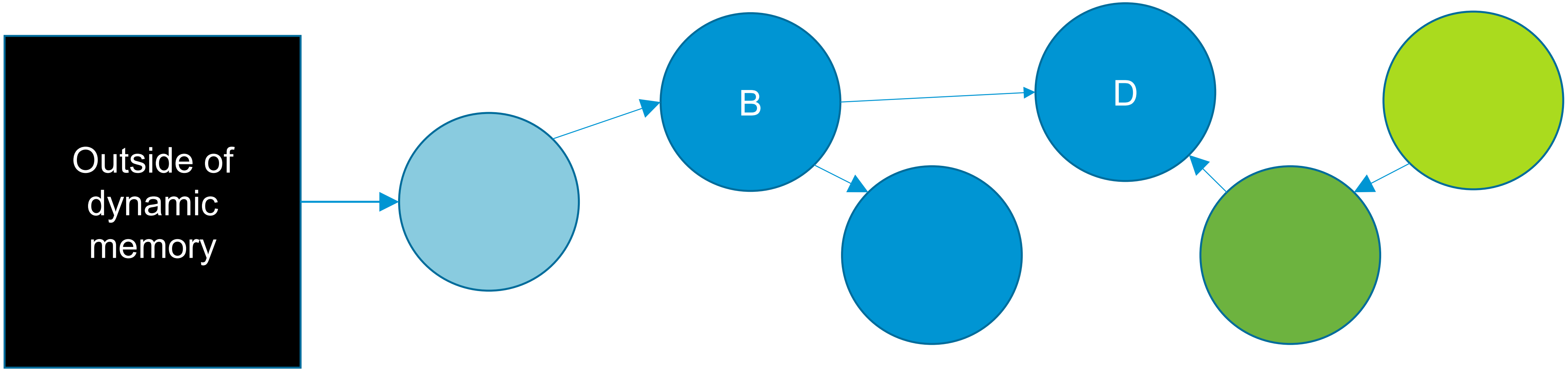
- A **used allocation** is considered an **anchor point** if it is directly referenced from **outside of dynamic memory**
- A **used allocation** is considered to be **anchored** if it is an **anchor point** or is referenced by an **anchored allocation**
- A **used allocation** that is not **anchored** is considered to be **leaked**
- A **leaked allocation** that is not referenced by another allocation is considered to be **unreferenced**



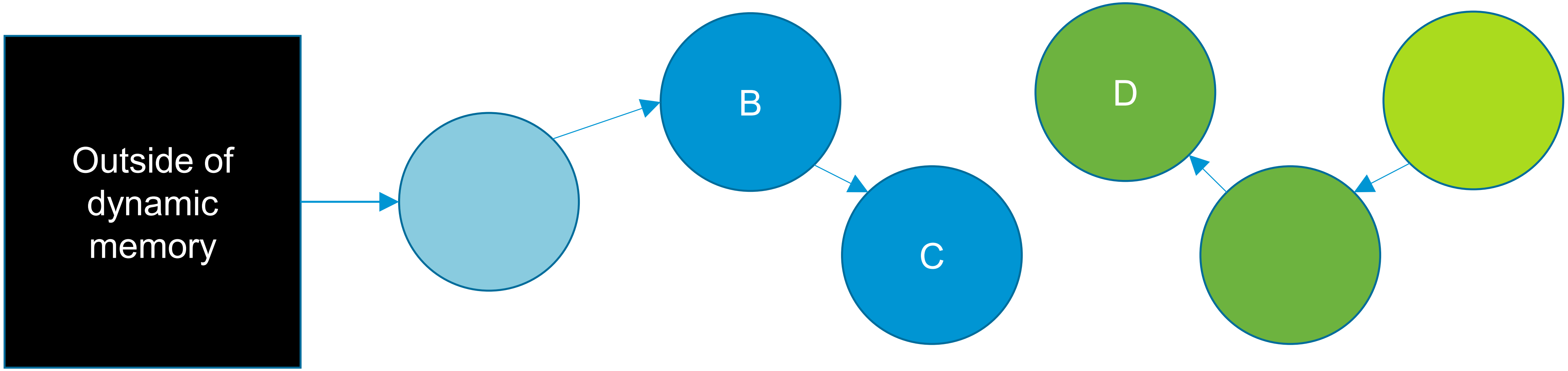
What Happens if a False Reference is Added From B to D?



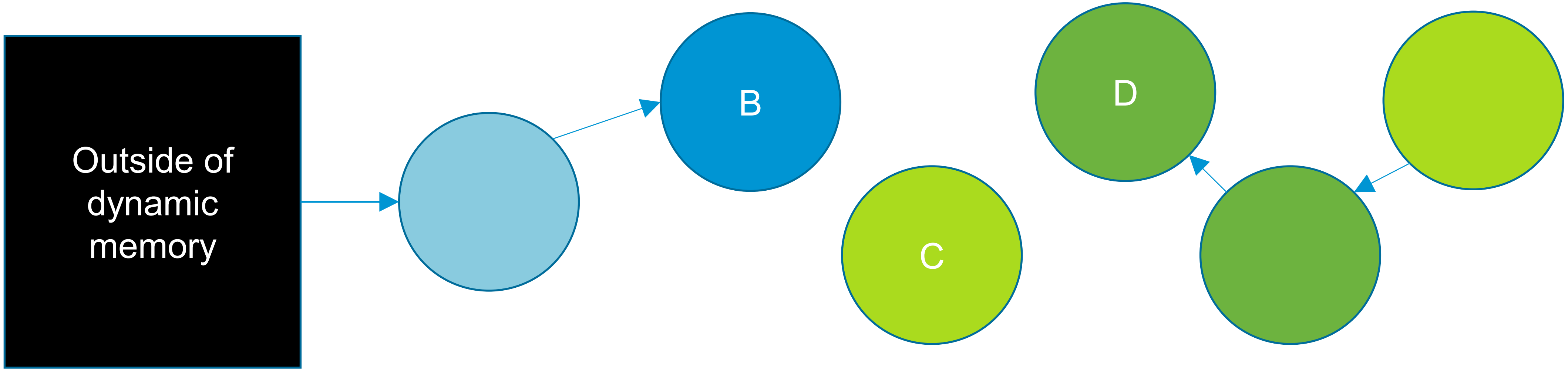
What Happens if a False Reference is Added From B to D?



What Happens if the Reference from B to C is Missed?

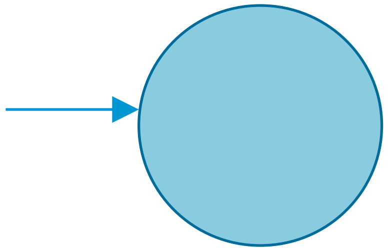


What Happens if the Reference from B to C is Missed?



What CHAP Considers to be References

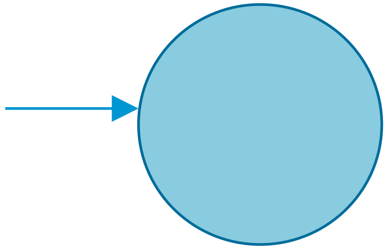
- A register associated with some thread contains a (not necessarily live) pointer p to some part of an allocation
- A pointer-sized range of memory (but constrained to be on a pointer sized boundary) contains a (not necessarily) live pointer p to some part of an allocation



-

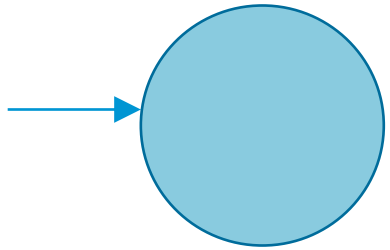
Some Reasons for False References Under CHAP

- Misinterpretation of liveness
  - Type not known
  - Failure to understand structure information for known type
  - Failure to understand liveness for known fields of a given class
  - Failure to understand liveness as a function of thread state
- Coincidence
  - Adjacent short integers
  - C-string

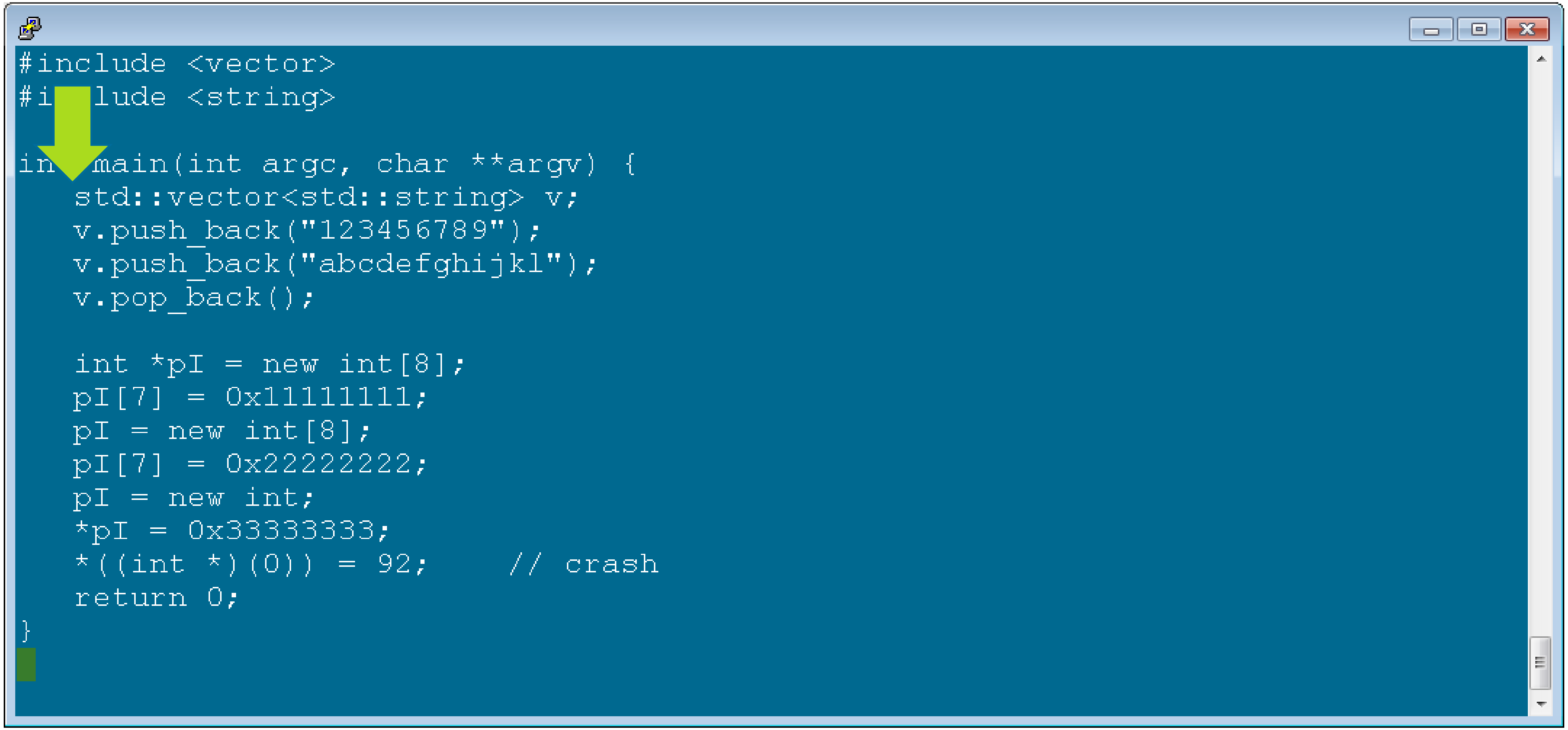


Some Reasons for Missed References Under CHAP

- Reference is from outside process
  - Fixable in future by allowing some way to recognize such allocations
- Reference is in the form f(p)
  - Fixable in future by modifying CHAP to be aware of f
- Reference is not aligned on a pointer-sized boundary
  - Fixable by relaxing alignment constraint, possibly configurably

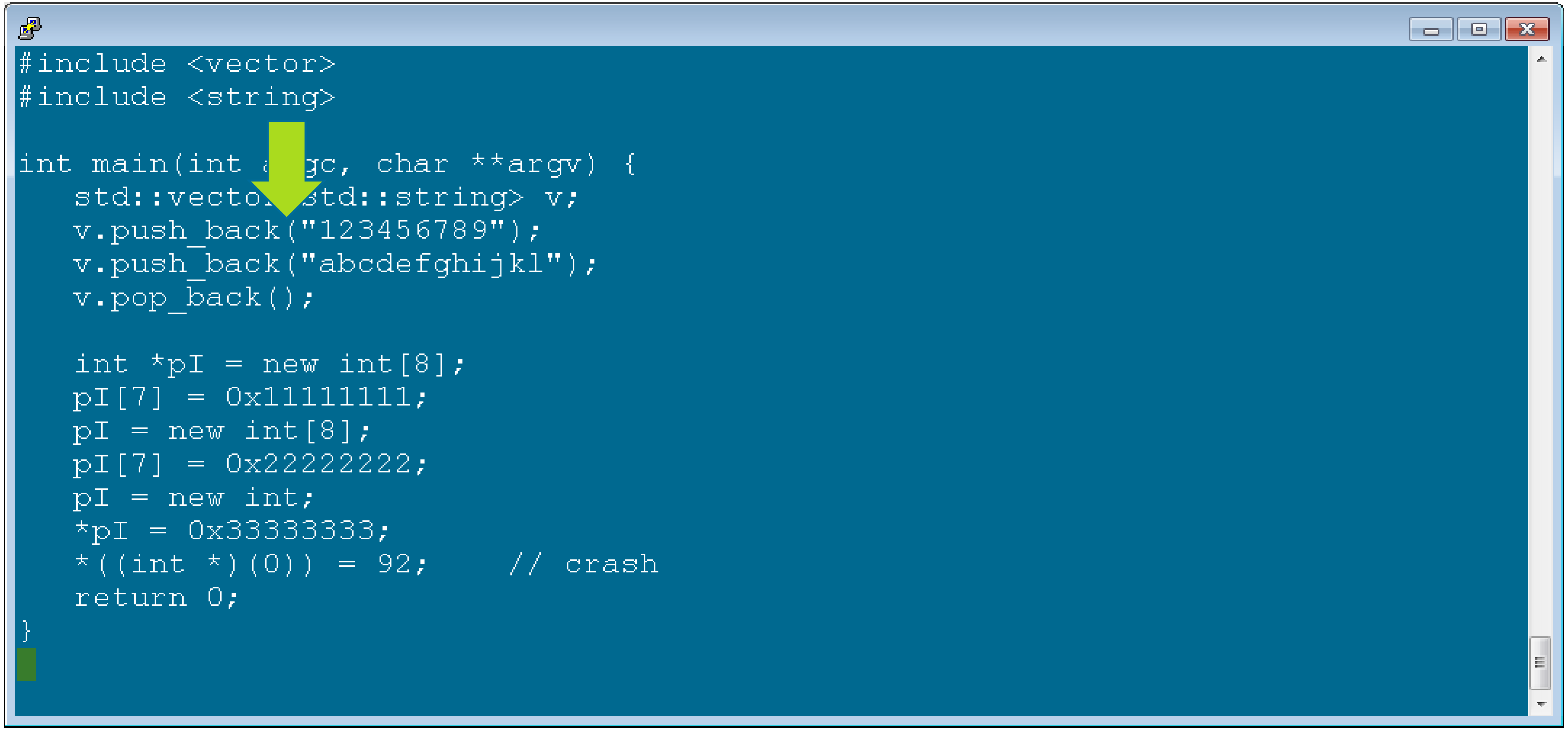


A Sample Program to Illustrate References



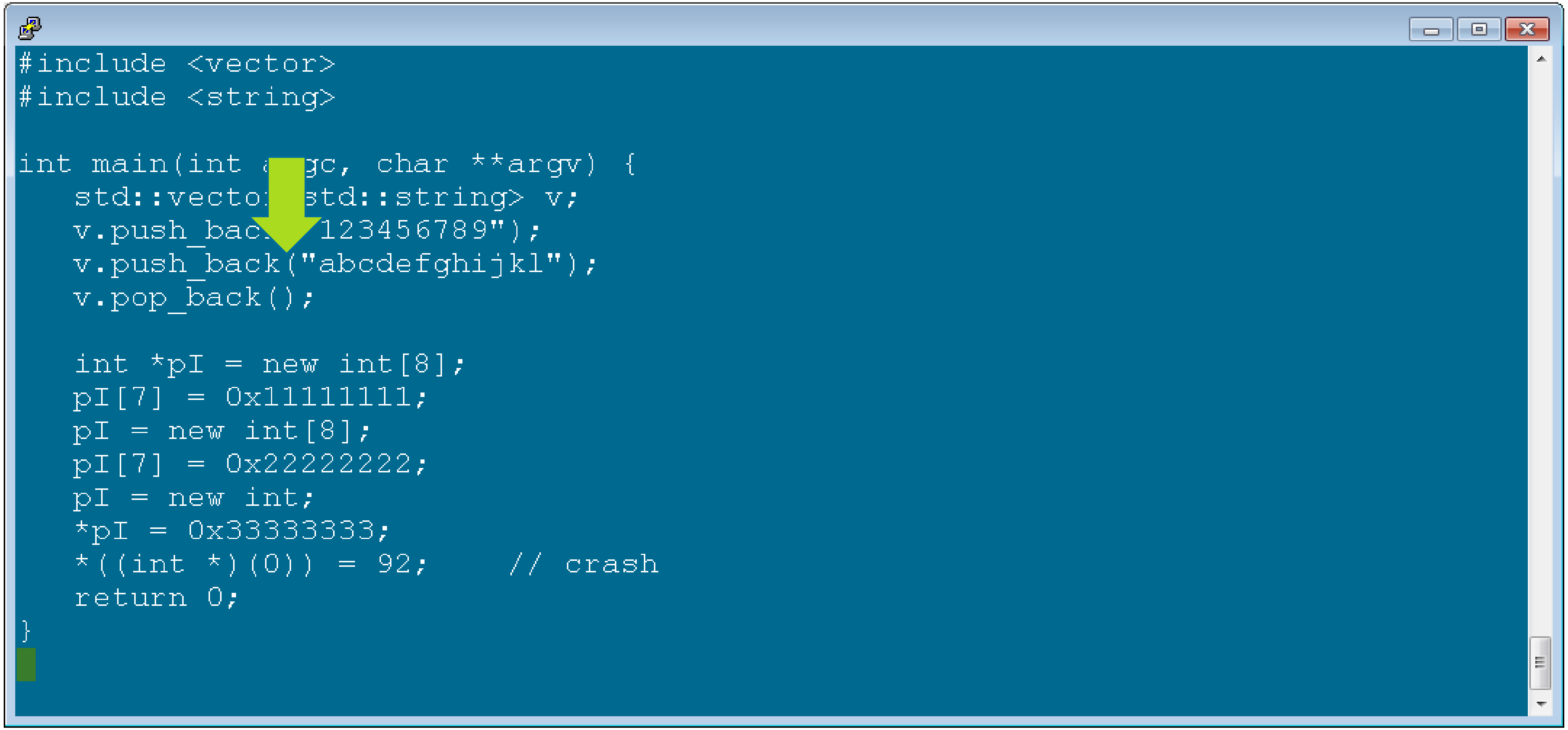
```
#include <vector>
#include <string>

int main(int argc, char **argv) {
   std::vector<std::string> v;
   v.push_back("123456789");
   v.push_back("abcdefghijkl");
   v.pop_back();

   int *pI = new int[8];
   pI[7] = 0x11111111;
   pI = new int[8];
   pI[7] = 0x22222222;
   pI = new int;
   *pI = 0x33333333;
   *((int *)(0)) = 92;    // crash
   return 0;
}
```

A Sample Program to Illustrate References



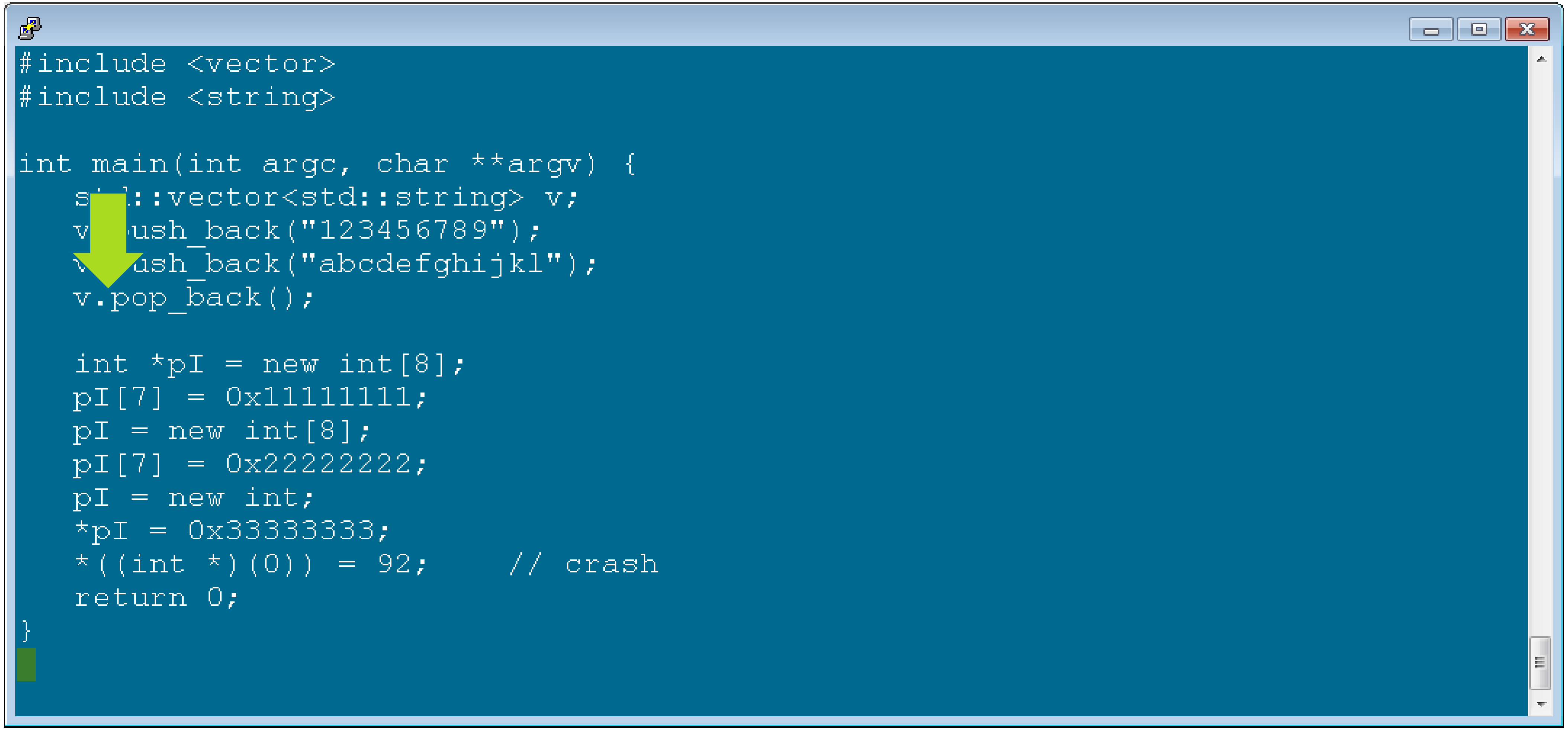
```
#include <vector>
#include <string>

int main(int argc, char **argv) {
   std::vector<std::string> v;
   v.push_back("123456789");
   v.push_back("abcdefghijkl");
   v.pop_back();

   int *pI = new int[8];
   pI[7] = 0x11111111;
   pI = new int[8];
   pI[7] = 0x22222222;
   pI = new int;
   *pI = 0x33333333;
   *((int *)(0)) = 92;     // crash
   return 0;
}
```

A Sample Program to Illustrate References



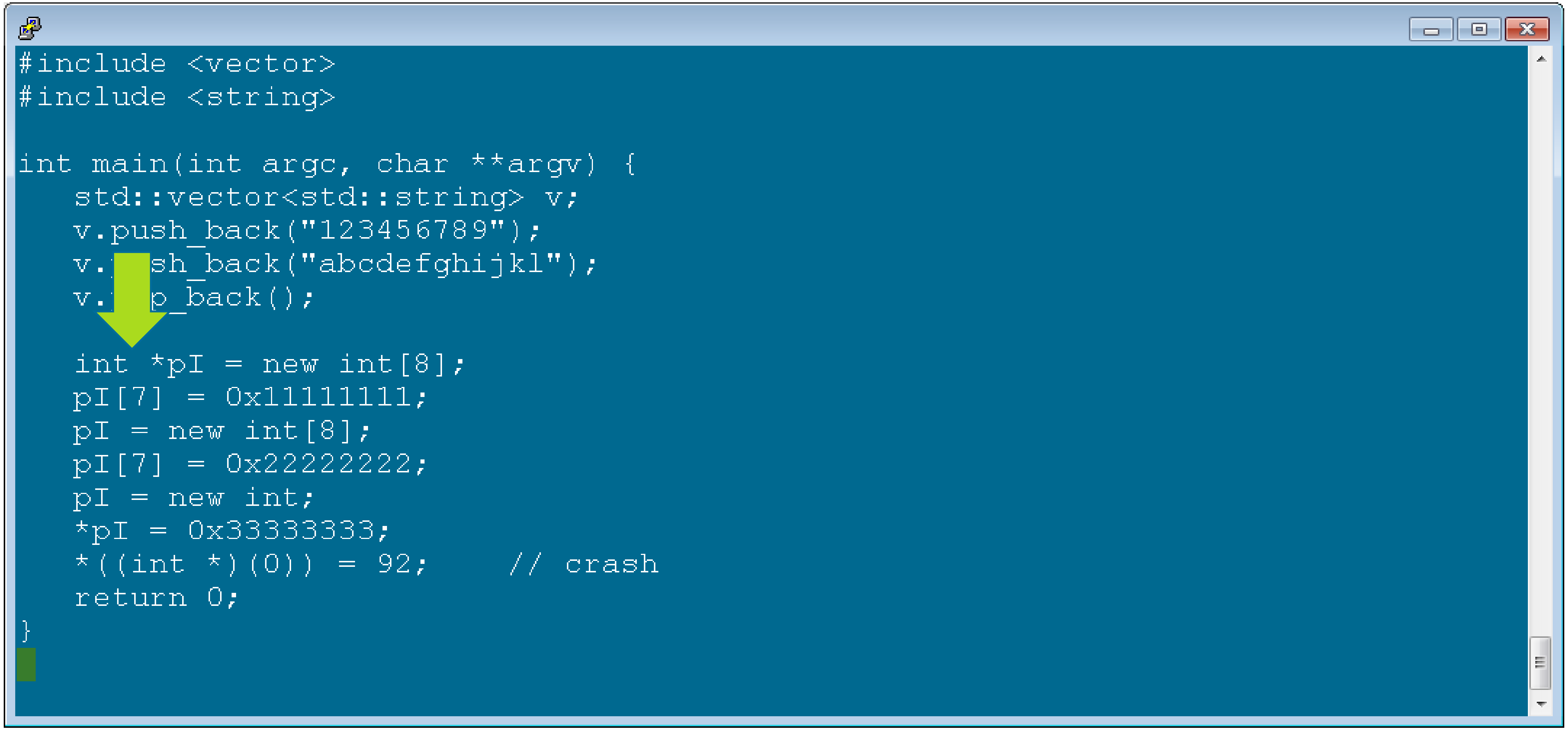
```
#include <vector>
#include <string>

int main(int argc, char **argv) {
   std::vector<std::string> v;
   v.push_back("123456789");
   v.push_back("abcdefghijkl");
   v.pop_back();

   int *pI = new int[8];
   pI[7] = 0x11111111;
   pI = new int[8];
   pI[7] = 0x22222222;
   pI = new int;
   *pI = 0x33333333;
   *((int *)(0)) = 92;     // crash
   return 0;
}
```

A Sample Program to Illustrate References



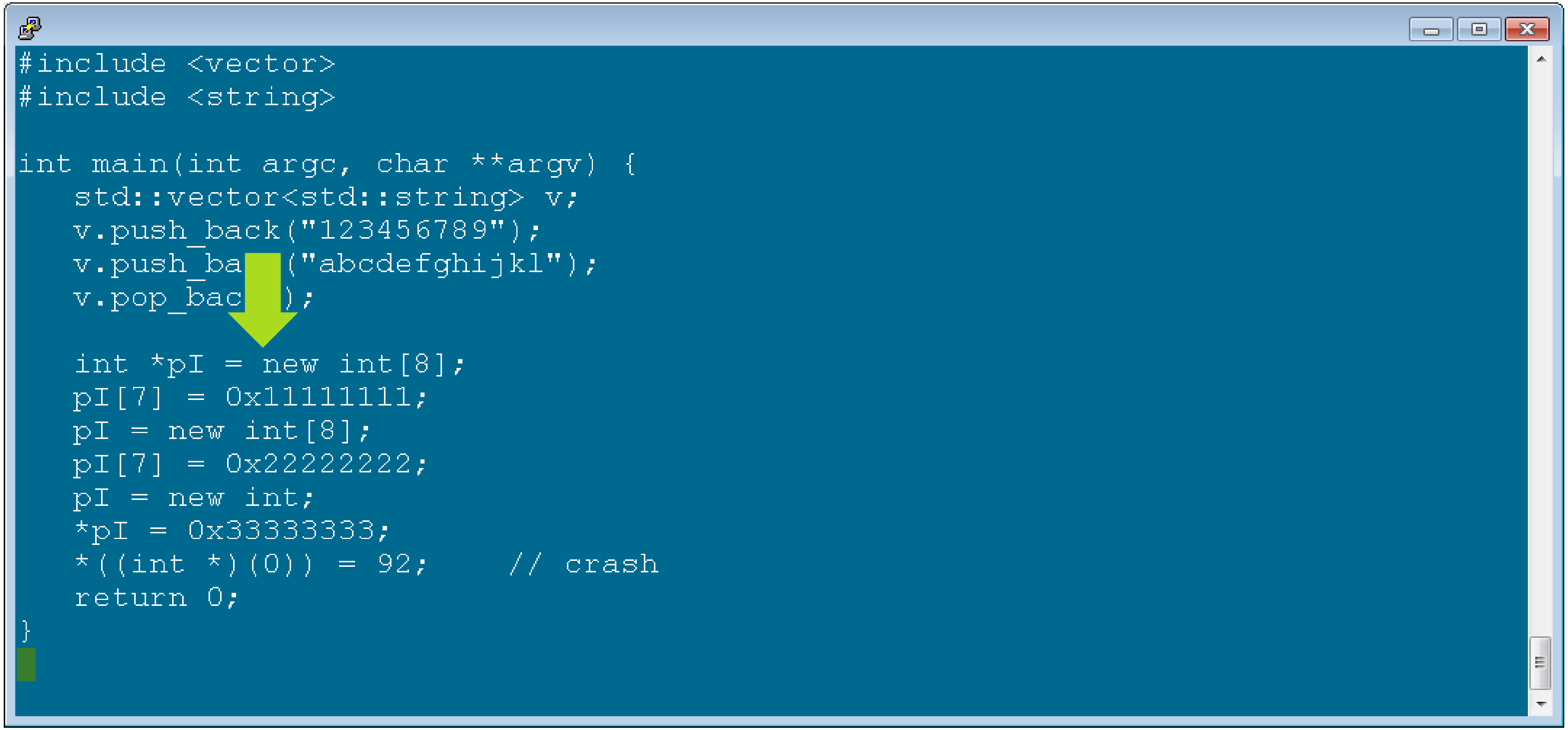
```
#include <vector>
#include <string>

int main(int argc, char **argv) {
   std::vector<std::string> v;
   v.push_back("123456789");
   v.push_back("abcdefghijkl");
   v.pop_back();

   int *pI = new int[8];
   pI[7] = 0x11111111;
   pI = new int[8];
   pI[7] = 0x22222222;
   pI = new int;
   *pI = 0x33333333;
   *((int *)(0)) = 92;    // crash
   return 0;
}
```

A Sample Program to Illustrate References



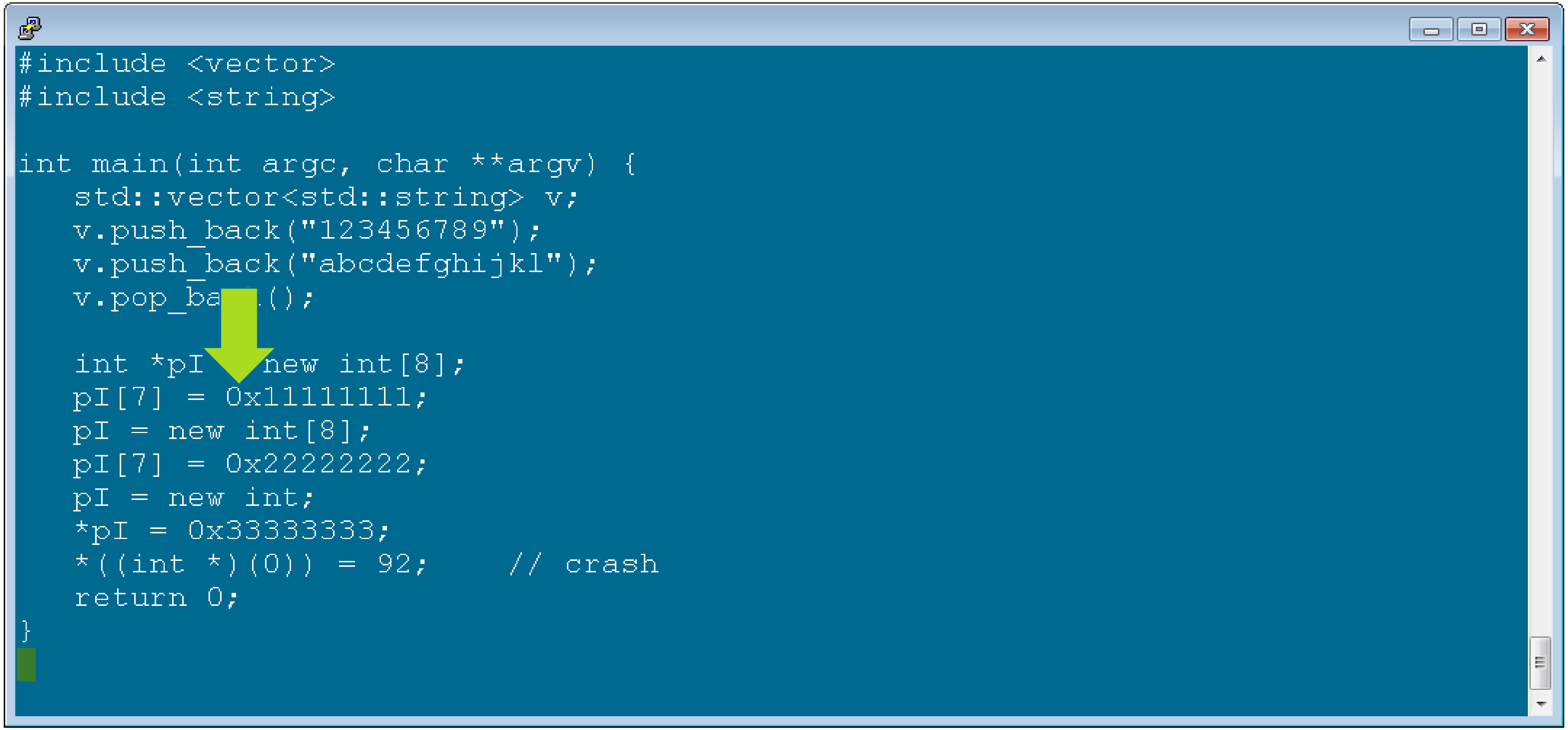
```
#include <vector>
#include <string>

int main(int argc, char **argv) {
   std::vector<std::string> v;
   v.push_back("123456789");
   v.push_back("abcdefghijkl");
   v.pop_back();

   int *pI = new int[8];
   pI[7] = 0x11111111;
   pI = new int[8];
   pI[7] = 0x22222222;
   pI = new int;
   *pI = 0x33333333;
   *((int *)(0)) = 92;     // crash
   return 0;
}
```

A Sample Program to Illustrate References



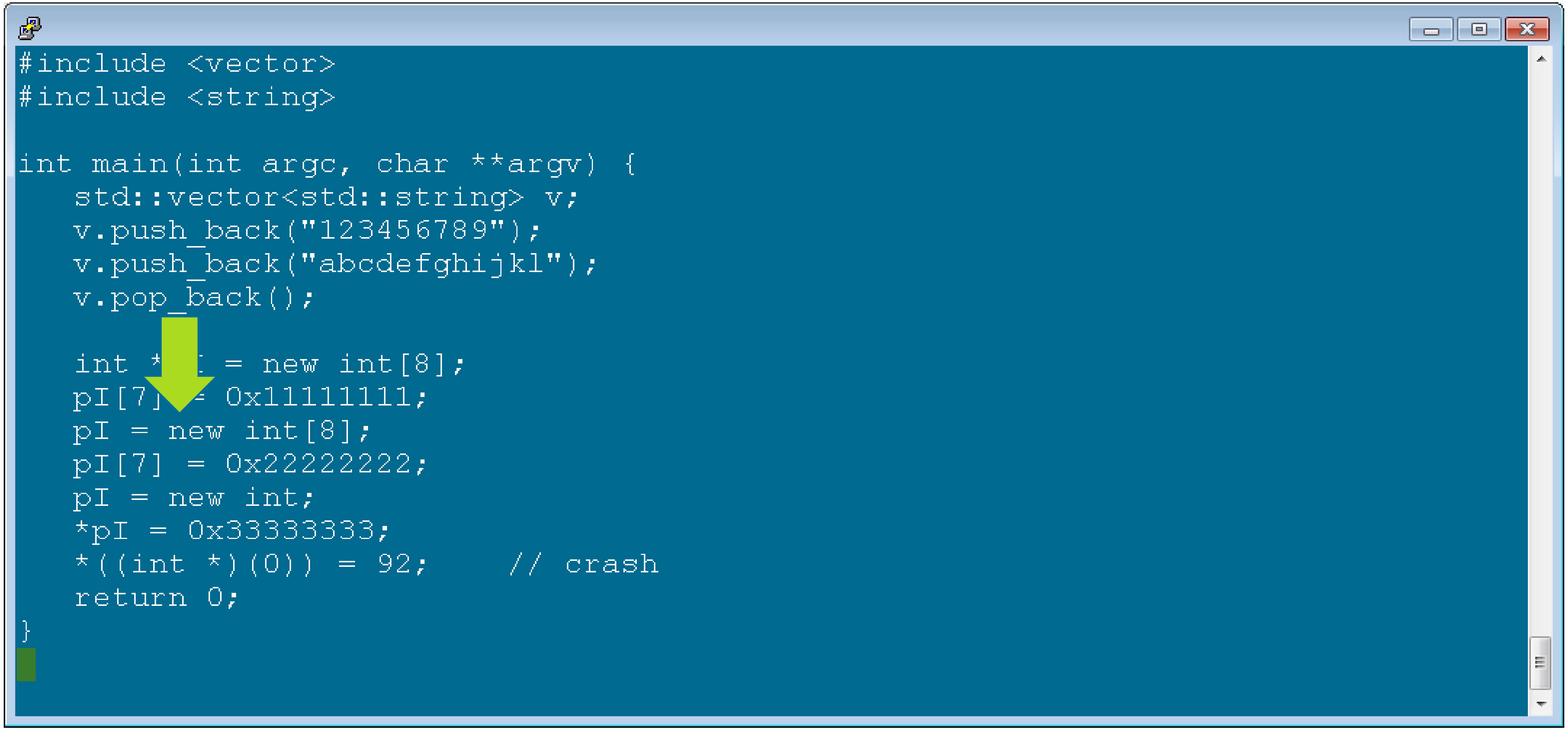
```
#include <vector>
#include <string>

int main(int argc, char **argv) {
   std::vector<std::string> v;
   v.push_back("123456789");
   v.push_back("abcdefghijkl");
   v.pop_back();

   int *pI = new int[8];
   pI[7] = 0x11111111;
   pI = new int[8];
   pI[7] = 0x22222222;
   pI = new int;
   *pI = 0x33333333;
   *((int *)(0)) = 92;     // crash
   return 0;
}
```

A Sample Program to Illustrate References



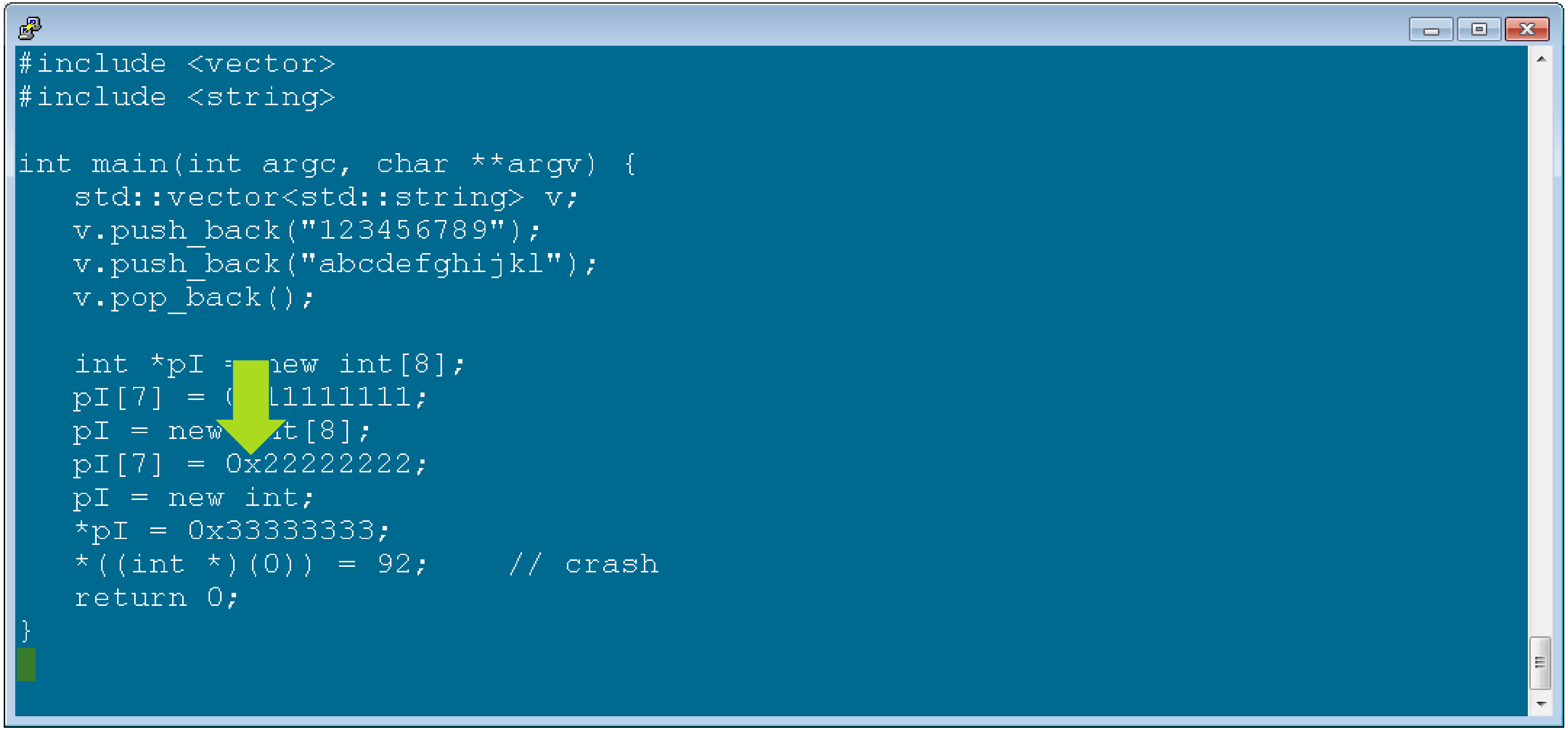
```
#include <vector>
#include <string>

int main(int argc, char **argv) {
   std::vector<std::string> v;
   v.push_back("123456789");
   v.push_back("abcdefghijkl");
   v.pop_back();

   int *pI = new int[8];
   pI[7] = 0x11111111;
   pI = new int[8];
   pI[7] = 0x22222222;
   pI = new int;
   *pI = 0x33333333;
   *((int *)(0)) = 92;     // crash
   return 0;
}
```

A Sample Program to Illustrate References



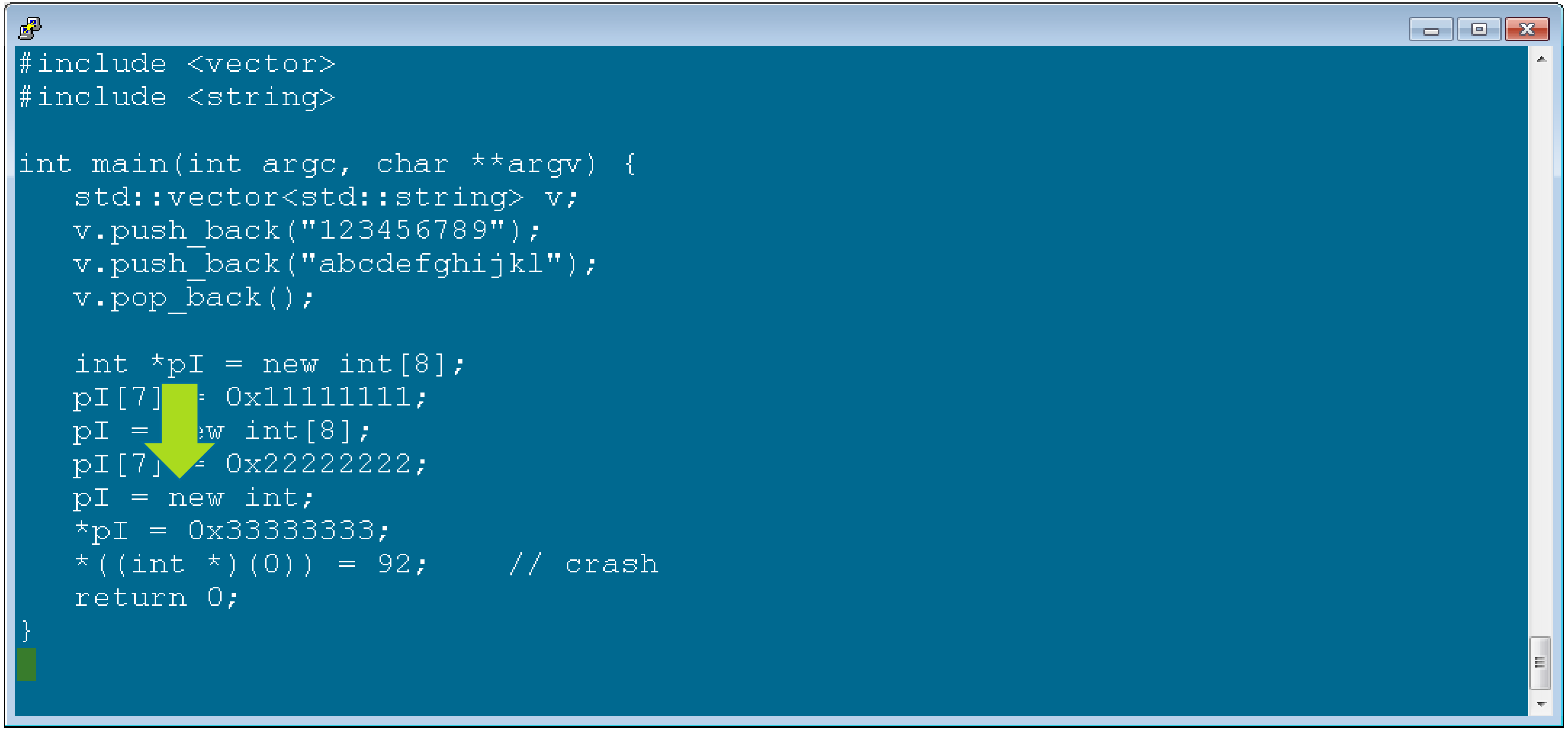
```
#include <vector>
#include <string>

int main(int argc, char **argv) {
   std::vector<std::string> v;
   v.push_back("123456789");
   v.push_back("abcdefghijkl");
   v.pop_back();

   int *pI = new int[8];
   pI[7] = 0x11111111;
   pI = new int[8];
   pI[7] = 0x22222222;
   pI = new int;
   *pI = 0x33333333;
   *((int *)(0)) = 92;     // crash
   return 0;
}
```

A Sample Program to Illustrate References



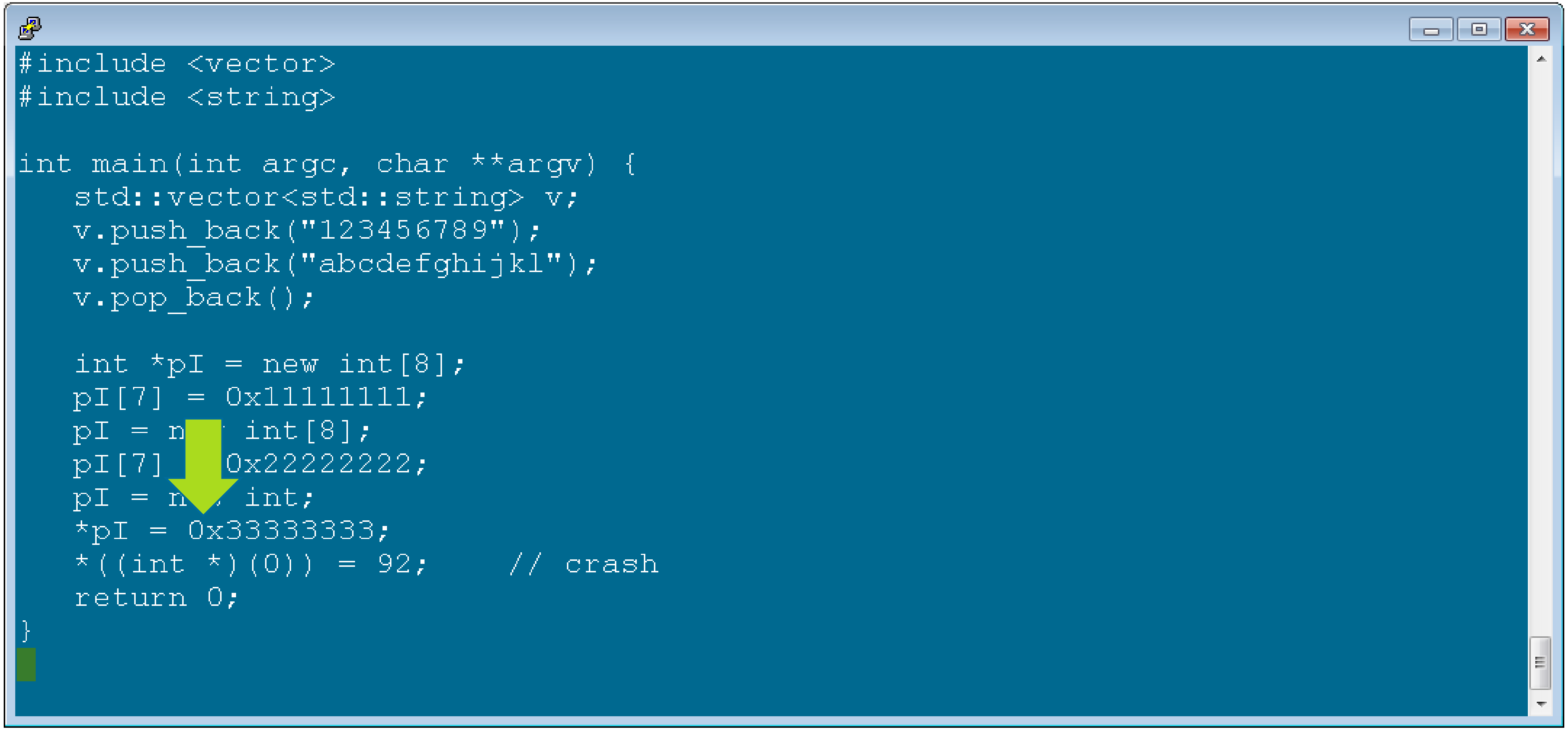
```
#include <vector>
#include <string>

int main(int argc, char **argv) {
   std::vector<std::string> v;
   v.push_back("123456789");
   v.push_back("abcdefghijkl");
   v.pop_back();

   int *pI = new int[8];
   pI[7] = 0x11111111;
   pI = new int[8];
   pI[7] = 0x22222222;
   pI = new int;
   *pI = 0x33333333;
   *((int *)(0)) = 92;     // crash
   return 0;
}
```

A Sample Program to Illustrate References



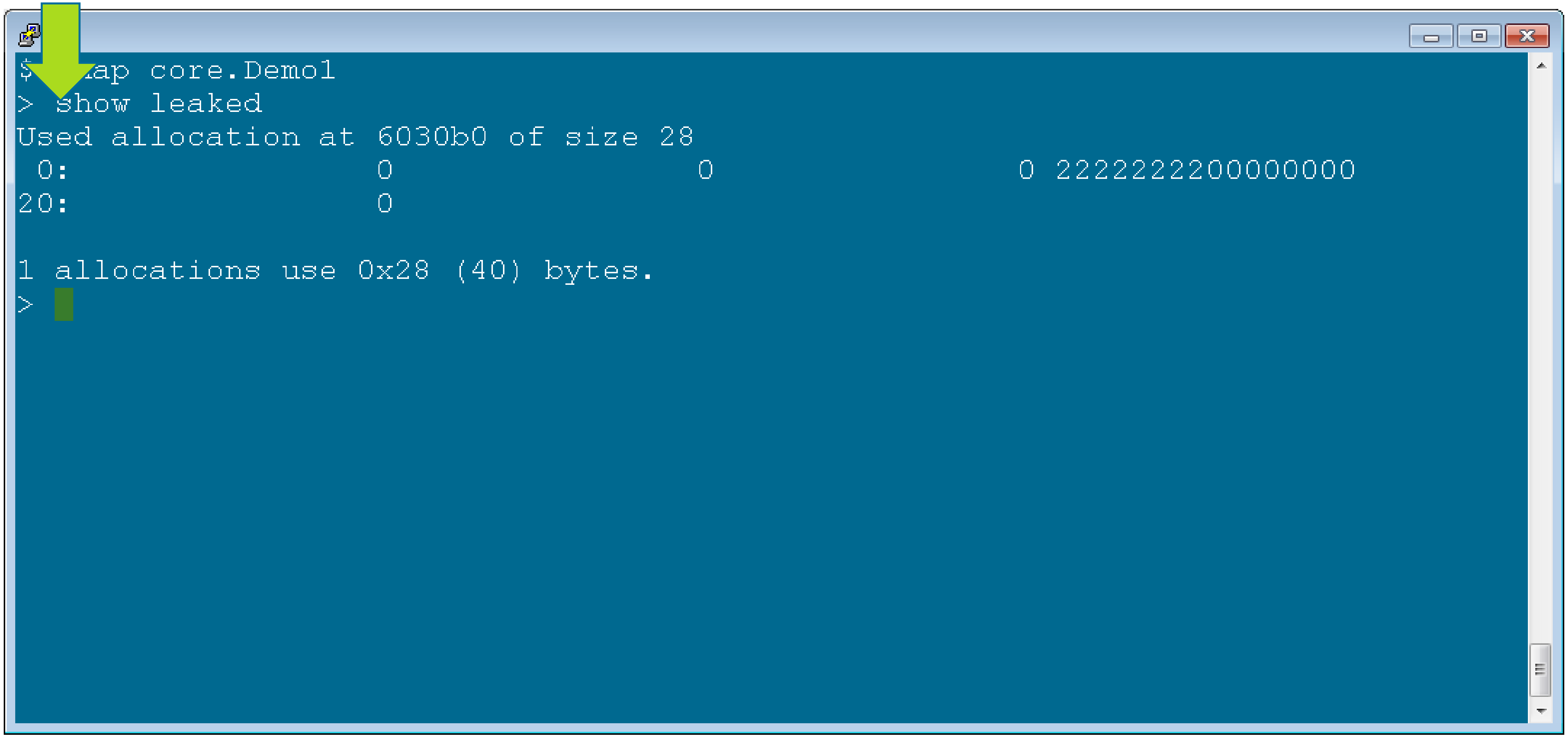
```
#include <vector>
#include <string>

int main(int argc, char **argv) {
   std::vector<std::string> v;
   v.push_back("123456789");
   v.push_back("abcdefghijkl");
   v.pop_back();

   int *pI = new int[8];
   pI[7] = 0x11111111;
   pI = new int[8];
   pI[7] = 0x22222222;
   pI = new int;
   *pI = 0x33333333;
   *((int *)(0)) = 92;     // crash
   return 0;
}
```

A Sample Program to Illustrate References



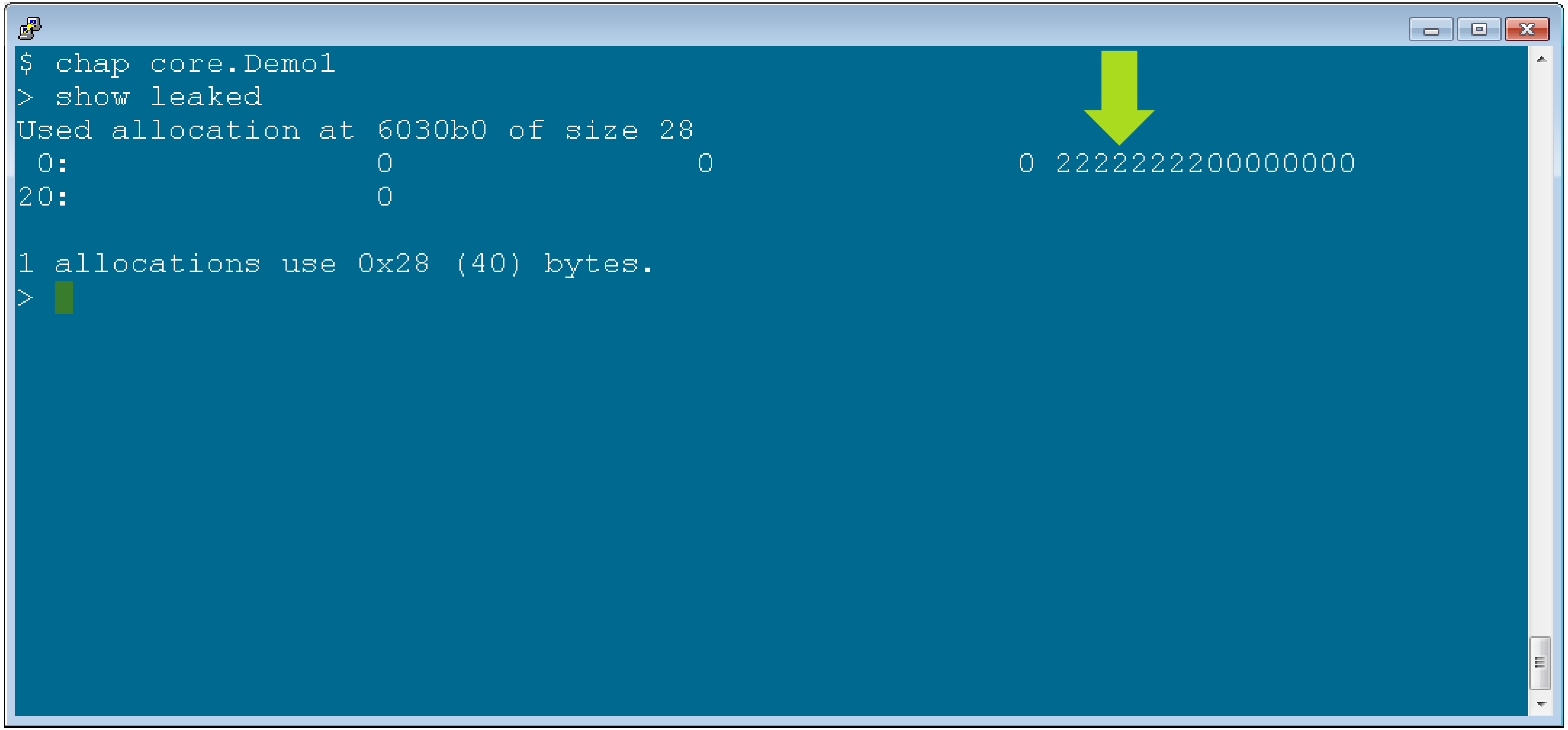
```
#include <vector>
#include <string>

int main(int argc, char **argv) {
   std::vector<std::string> v;
   v.push_back("123456789");
   v.push_back("abcdefghijkl");
   v.pop_back();

   int *pI = new int[8];
   pI[7] = 0x11111111;
   pI = new int[8];
   pI[7] = 0x22222222;
   pI = new int;
   *pI = 0x33333333;
   *((int *)(0)) = 92;     // crash
   return 0;
}
```

Showing (some of the) Leaked Allocations



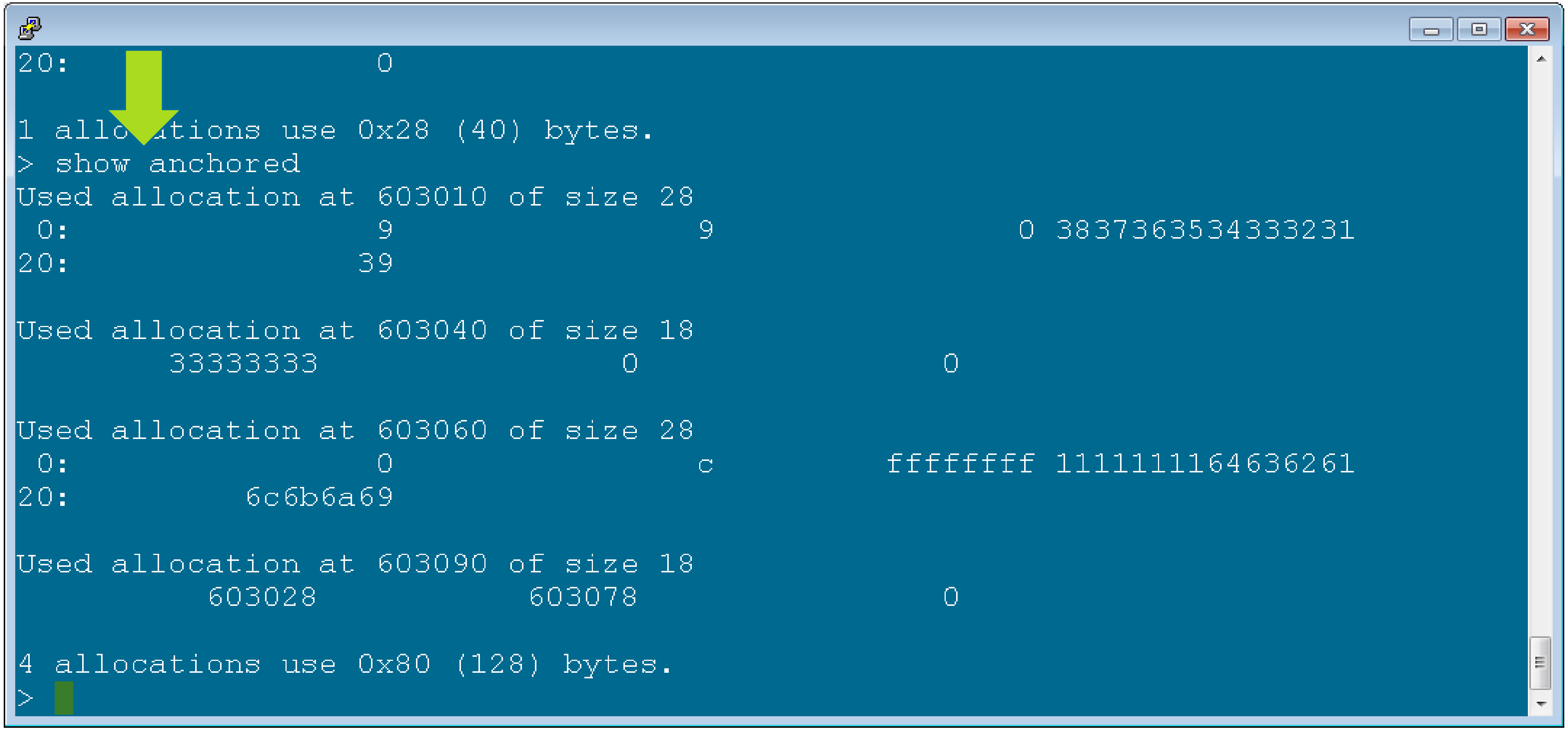
```
$ chap core.Demo1
> show leaked
Used allocation at 6030b0 of size 28
 0:                0                0                0 2222222200000000
20:                0

1 allocations use 0x28 (40) bytes.
> 
```

Showing (some of the) Leaked Allocations



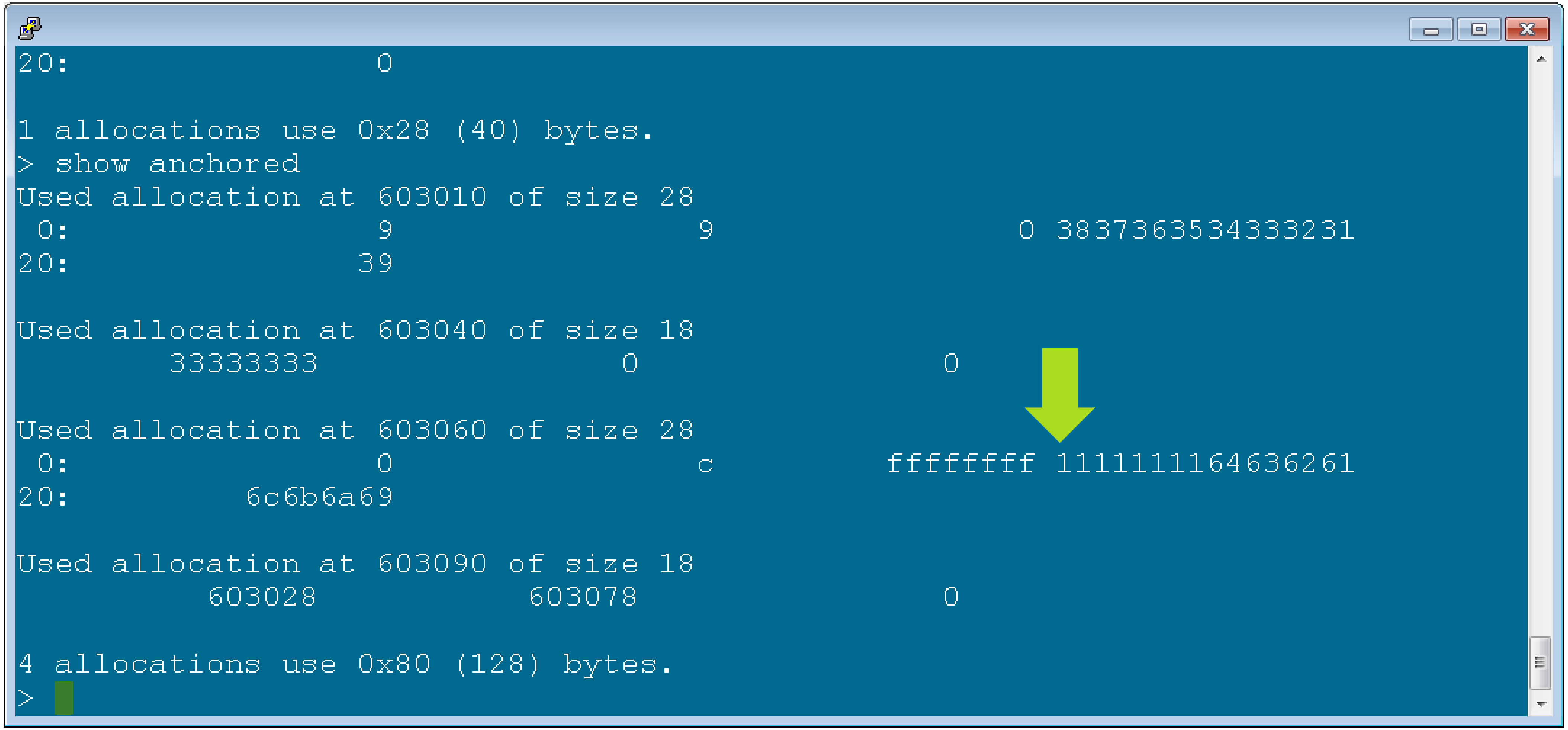
```
$ chap core.Demo1
> show leaked
Used allocation at 6030b0 of size 28
 0:                0                0                0 2222222200000000
20:                0

1 allocations use 0x28 (40) bytes.
> 
```

Showing (too many) Anchored Allocations



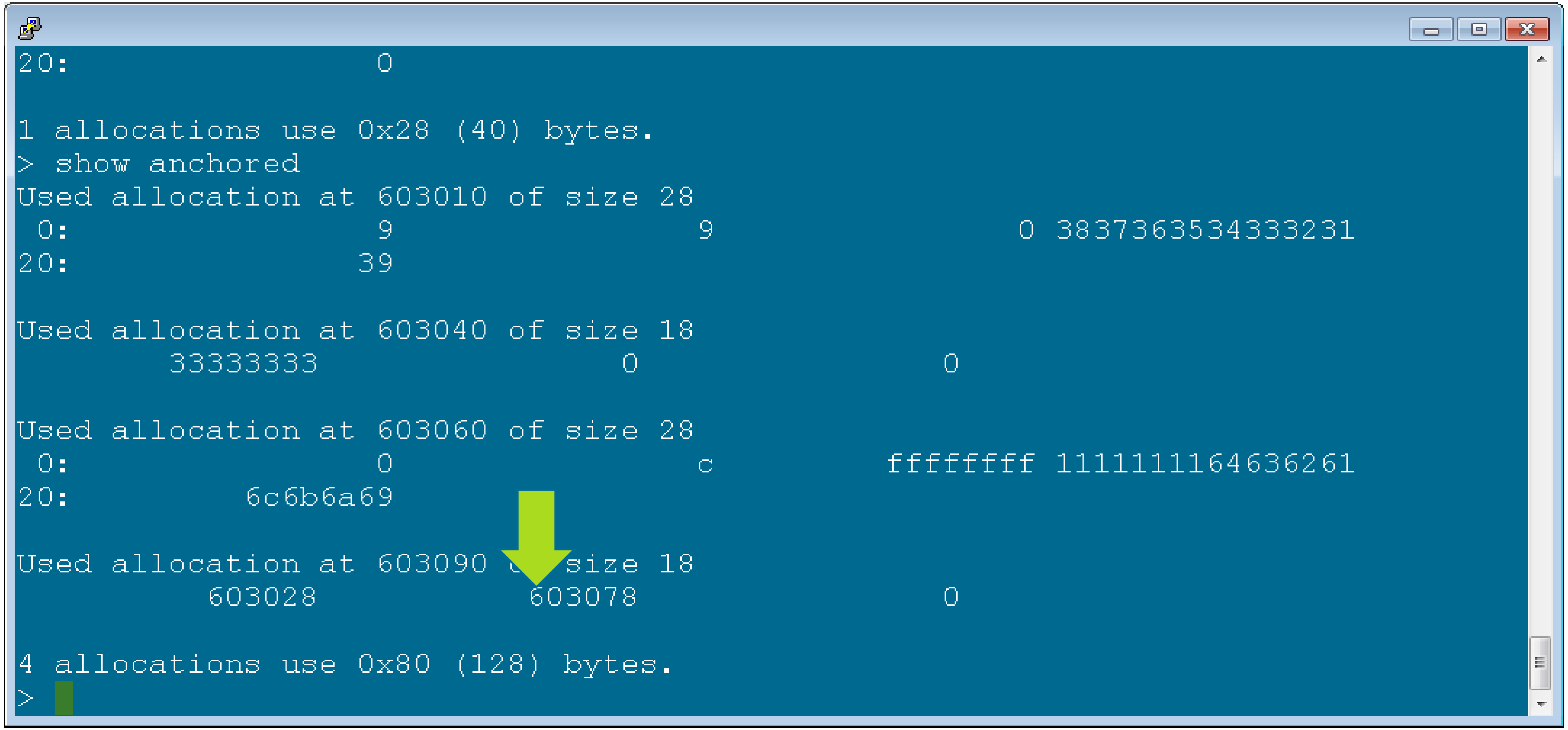
```
20:                0

1 allocations use 0x28 (40) bytes.
> show anchored
Used allocation at 603010 of size 28
 0:                9                9                0 3837363534333231
20:               39

Used allocation at 603040 of size 18
        33333333                0                0

Used allocation at 603060 of size 28
 0:                0                c         ffffffff 1111111164636261
20:         6c6b6a69

Used allocation at 603090 of size 18
          603028           603078                0

4 allocations use 0x80 (128) bytes.
>
```

Showing (too many) Anchored Allocations



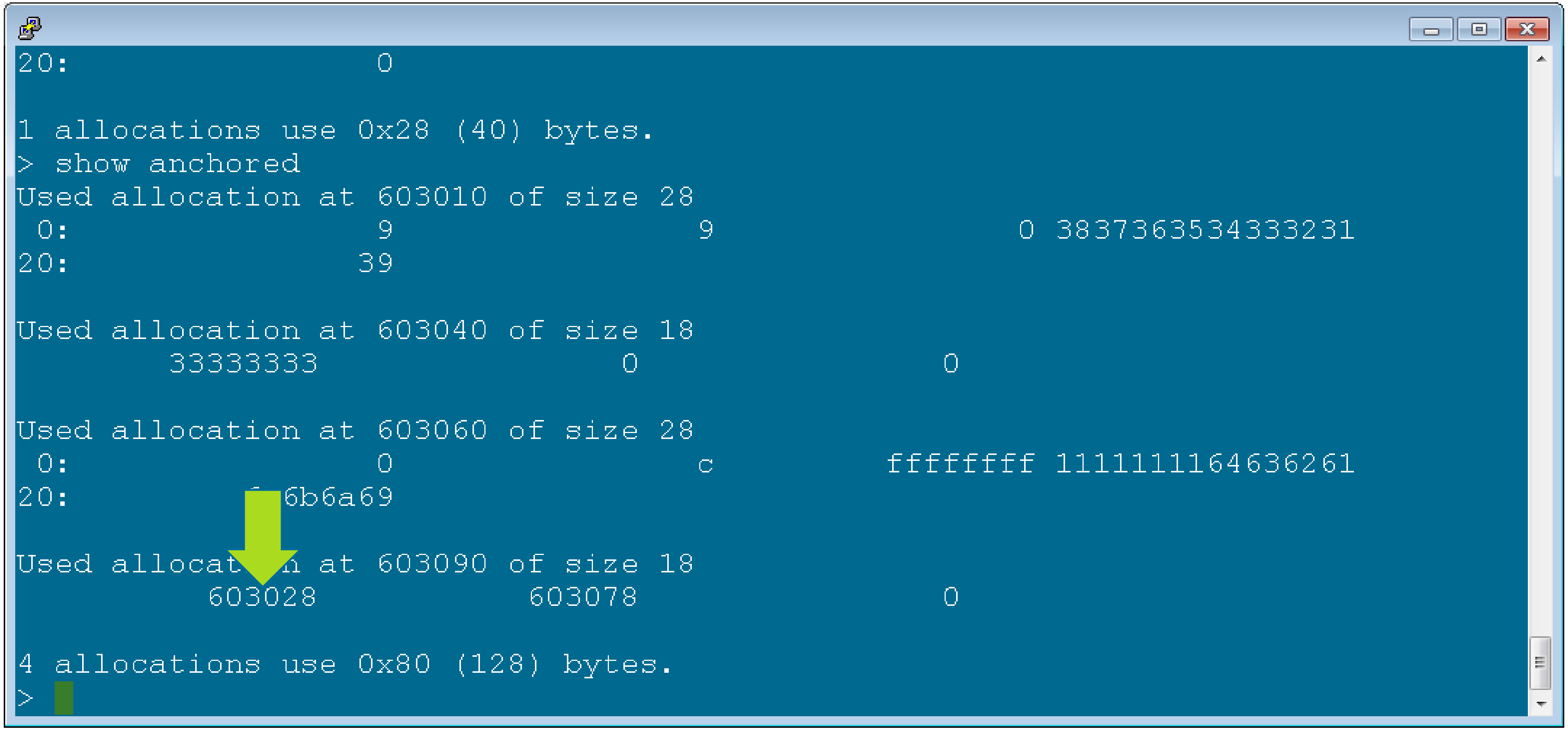
```
20:                0

1 allocations use 0x28 (40) bytes.
> show anchored
Used allocation at 603010 of size 28
 0:                9                9                0 3837363534333231
20:               39

Used allocation at 603040 of size 18
        33333333                0                0

Used allocation at 603060 of size 28
 0:                0                c         ffffffff 1111111164636261
20:         6c6b6a69

Used allocation at 603090 of size 18
          603028           603078                0

4 allocations use 0x80 (128) bytes.
>
```

Showing (too many) Anchored Allocations



```
20:                0

1 allocations use 0x28 (40) bytes.
> show anchored
Used allocation at 603010 of size 28
 0:                9                9                0 3837363534333231
20:               39

Used allocation at 603040 of size 18
        33333333                0                0

Used allocation at 603060 of size 28
 0:                0                c         ffffffff 1111111164636261
20:         6c6b6a69

Used allocation at 603090 of size 18
          603028           603078                0

4 allocations use 0x80 (128) bytes.
> 
```

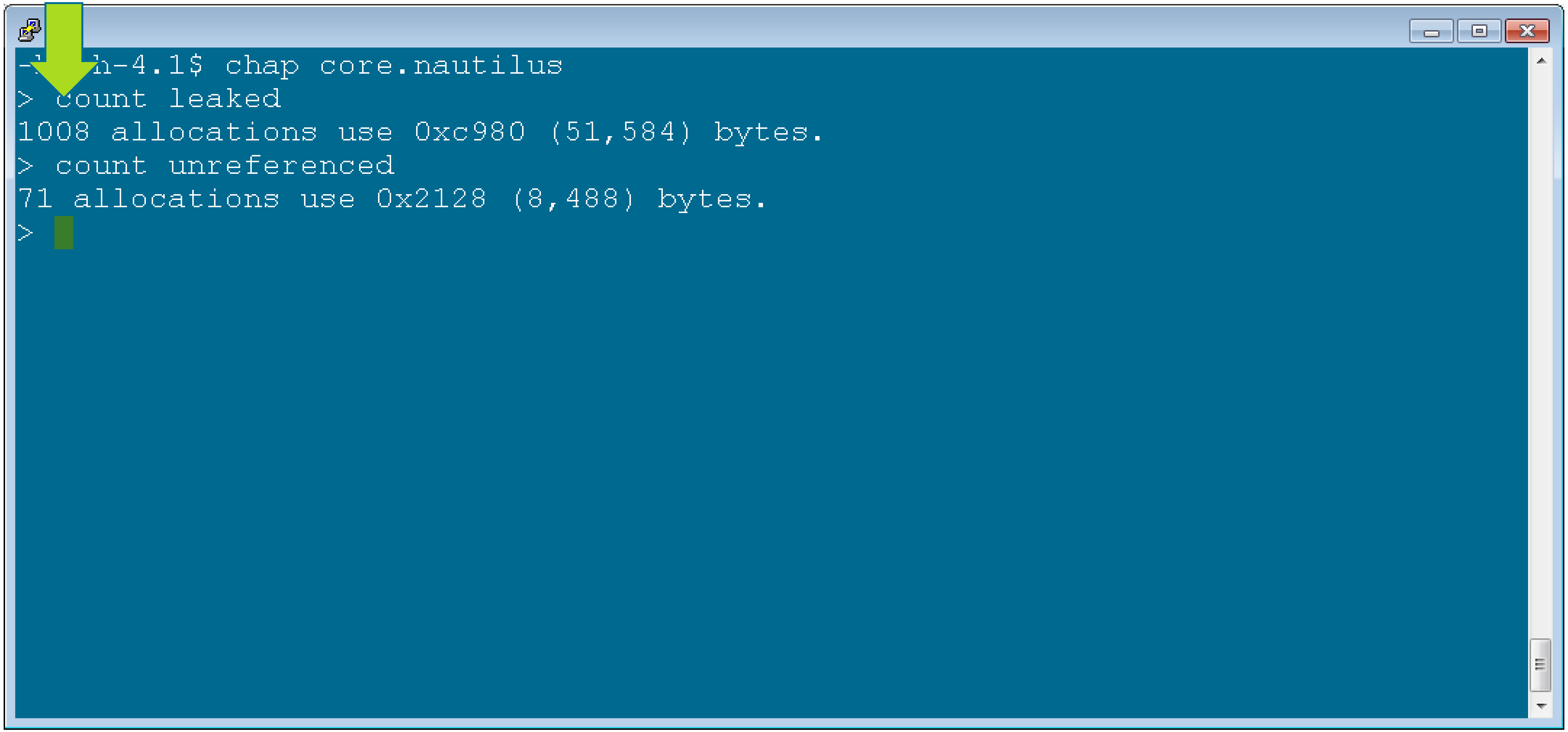
Showing (too many) Anchored Allocations



```
20:                0

1 allocations use 0x28 (40) bytes.
> show anchored
Used allocation at 603010 of size 28
 0:                9                9                0 3837363534333231
20:               39

Used allocation at 603040 of size 18
        33333333                0                0

Used allocation at 603060 of size 28
 0:                0                c         ffffffff 1111111164636261
20:             6b6a69

Used allocation at 603090 of size 18
          603028           603078                0

4 allocations use 0x80 (128) bytes.
> 
```

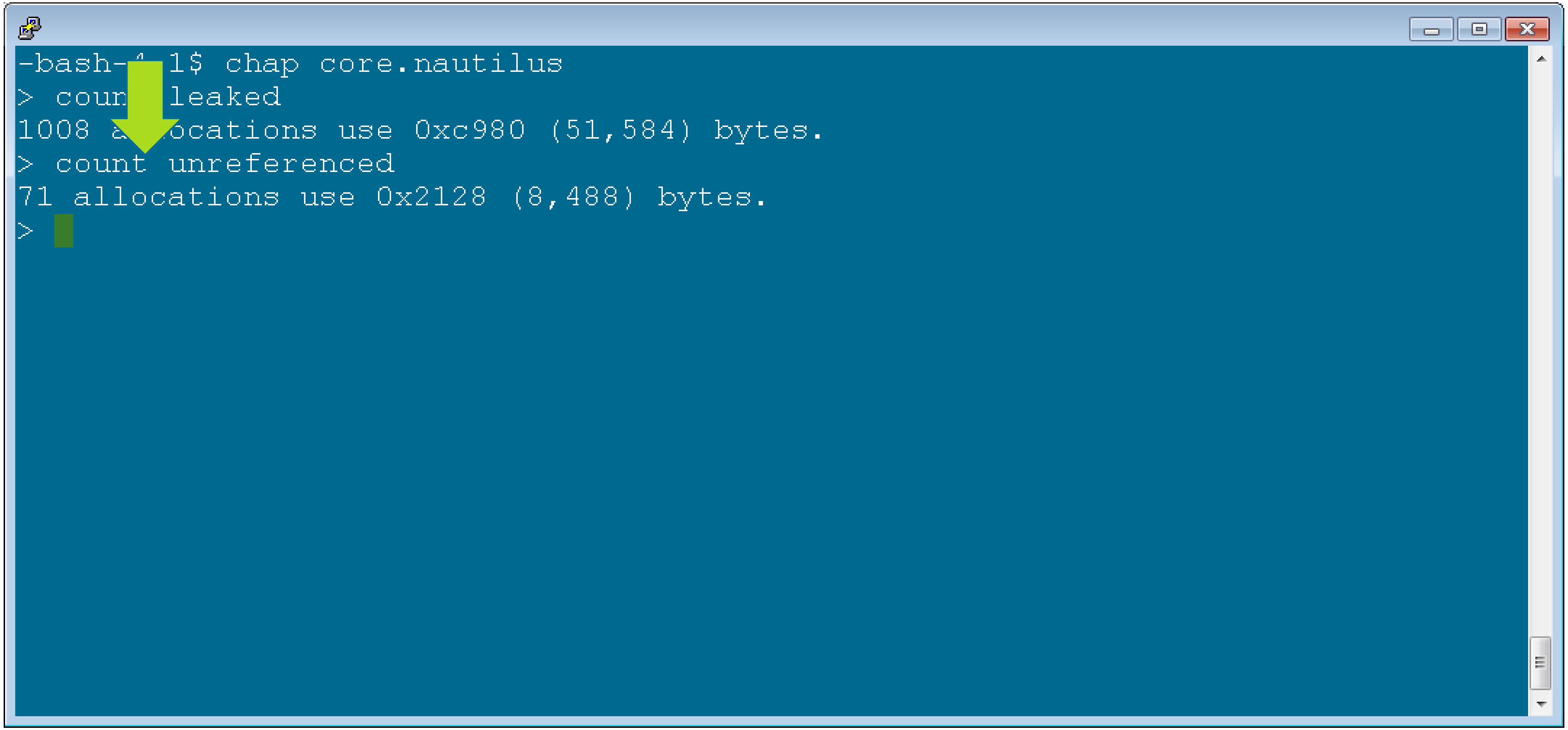
Using CHAP to Analyze Leaks

Checking for Leaks



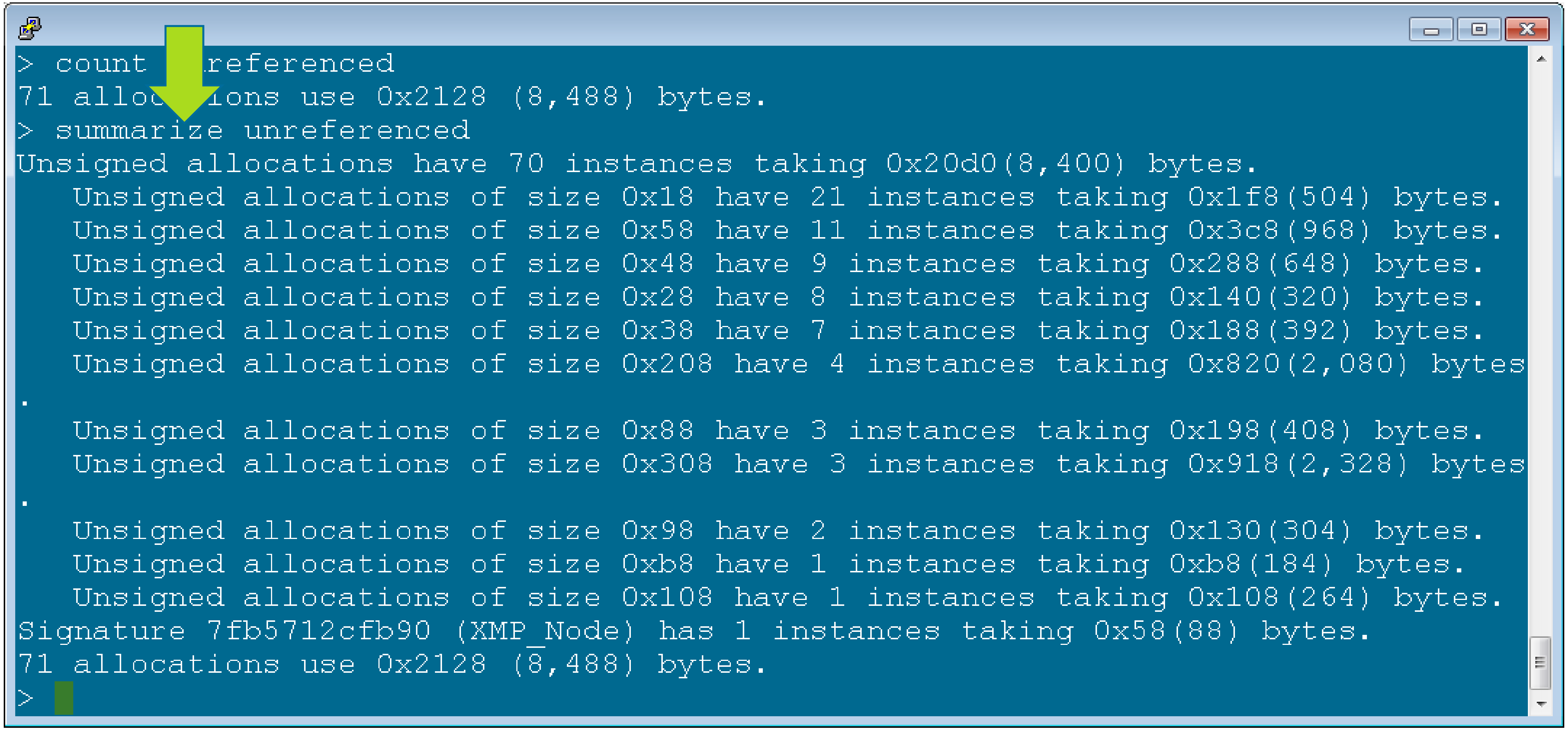
```
-bash-4.1$ chap core.nautilus
> count leaked
1008 allocations use 0xc980 (51,584) bytes.
> count unreferenced
71 allocations use 0x2128 (8,488) bytes.
> 
```

Checking for Leaks



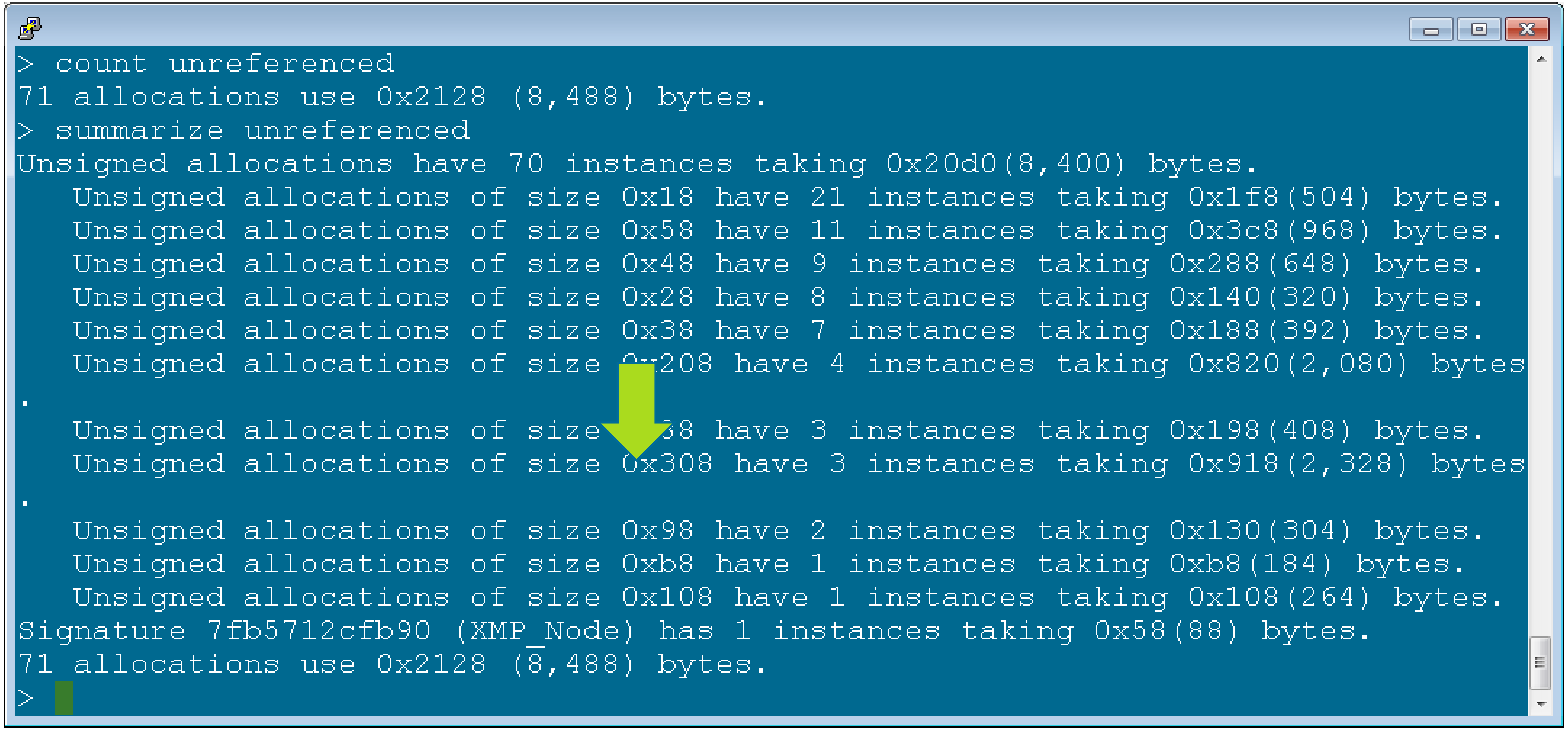
```
-bash-4.1$ chap core.nautilus
> count leaked
1008 allocations use 0xc980 (51,584) bytes.
> count unreferenced
71 allocations use 0x2128 (8,488) bytes.
>
```

Summarizing Unreferenced Allocations



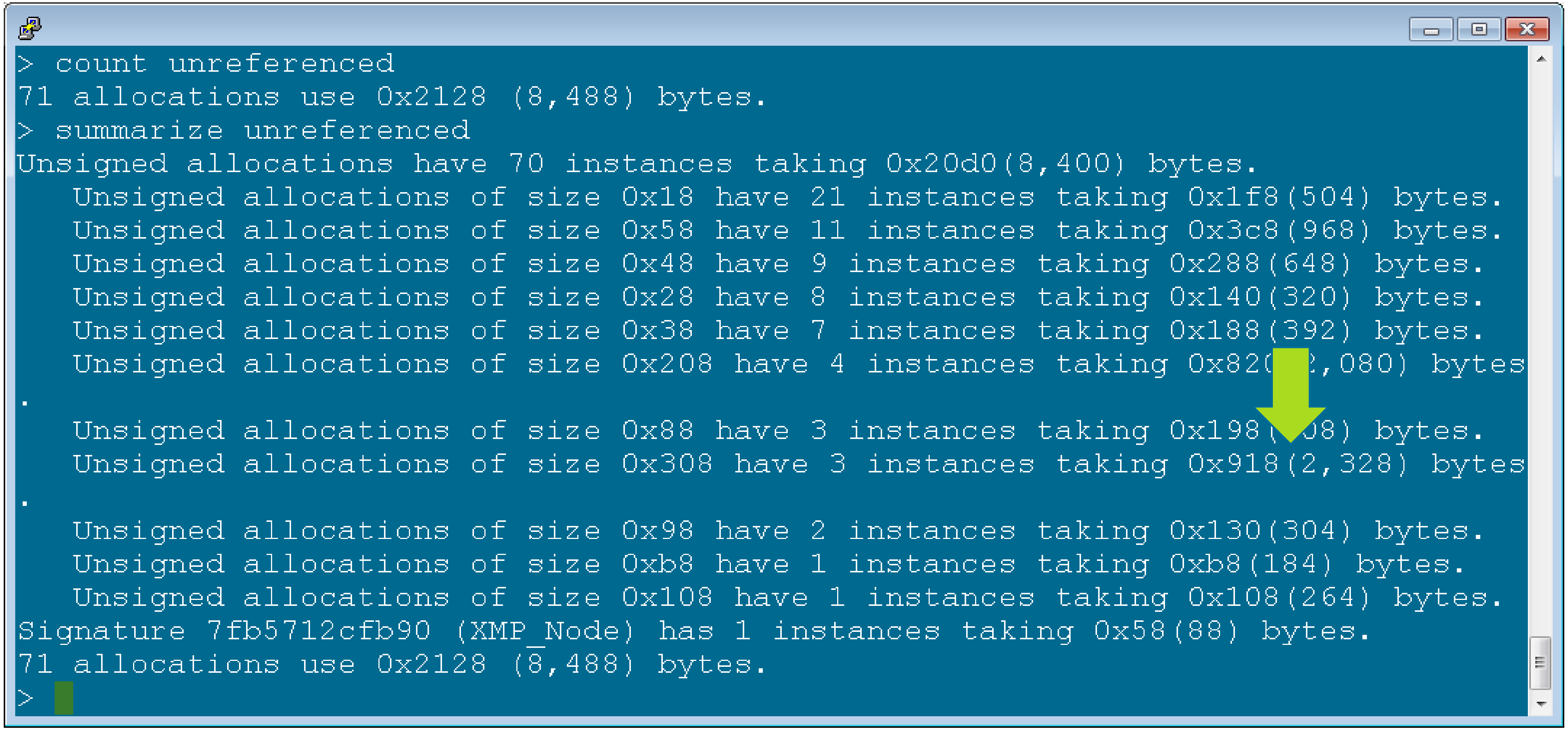
```
> count unreferenced
71 allocations use 0x2128 (8,488) bytes.
> summarize unreferenced
Unsigned allocations have 70 instances taking 0x20d0(8,400) bytes.
   Unsigned allocations of size 0x18 have 21 instances taking 0x1f8(504) bytes.
   Unsigned allocations of size 0x58 have 11 instances taking 0x3c8(968) bytes.
   Unsigned allocations of size 0x48 have 9 instances taking 0x288(648) bytes.
   Unsigned allocations of size 0x28 have 8 instances taking 0x140(320) bytes.
   Unsigned allocations of size 0x38 have 7 instances taking 0x188(392) bytes.
   Unsigned allocations of size 0x208 have 4 instances taking 0x820(2,080) bytes
.
   Unsigned allocations of size 0x88 have 3 instances taking 0x198(408) bytes.
   Unsigned allocations of size 0x308 have 3 instances taking 0x918(2,328) bytes
.
   Unsigned allocations of size 0x98 have 2 instances taking 0x130(304) bytes.
   Unsigned allocations of size 0xb8 have 1 instances taking 0xb8(184) bytes.
   Unsigned allocations of size 0x108 have 1 instances taking 0x108(264) bytes.
Signature 7fb5712cfb90 (XMP_Node) has 1 instances taking 0x58(88) bytes.
71 allocations use 0x2128 (8,488) bytes.
> 
```

Summarizing Unreferenced Allocations



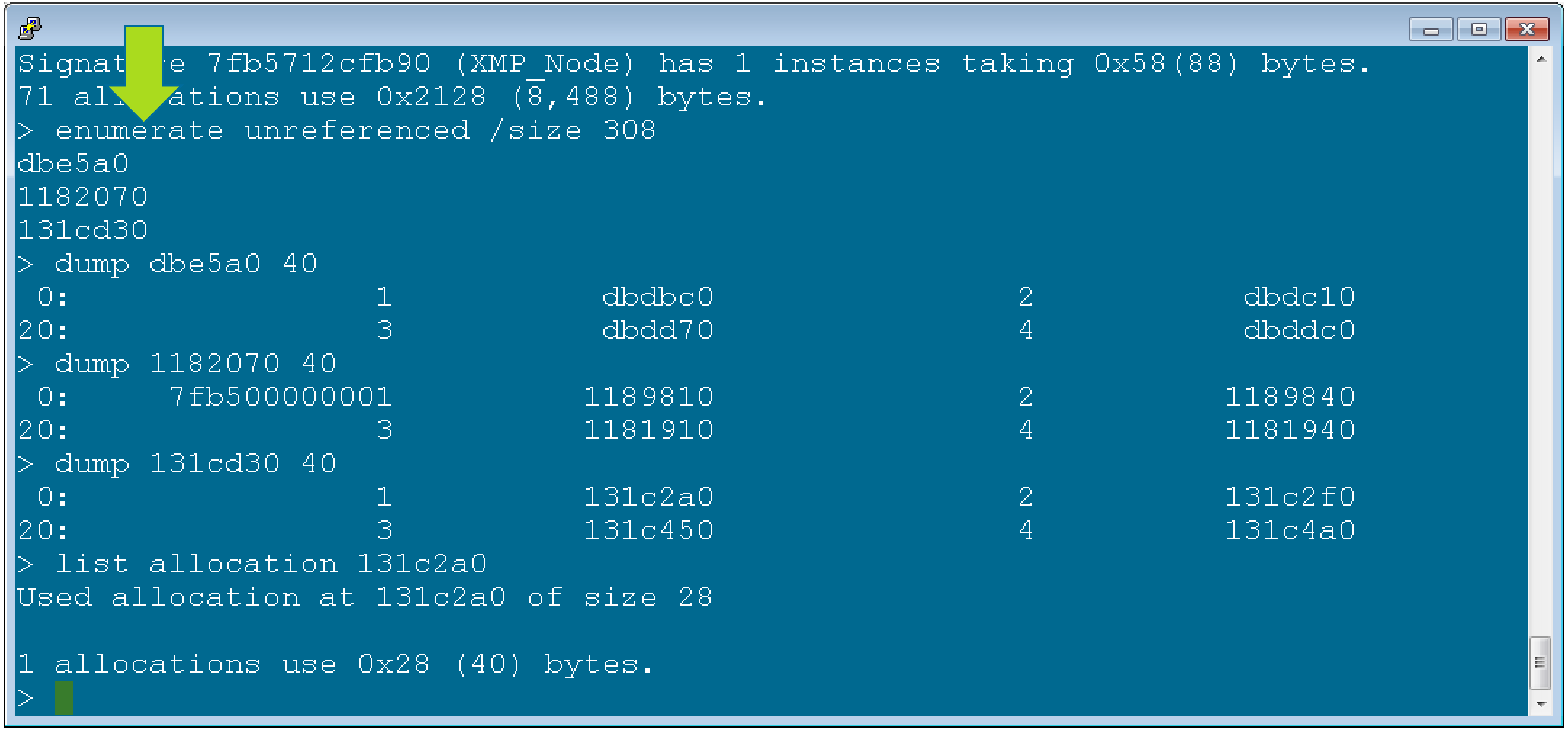
```
> count unreferenced
71 allocations use 0x2128 (8,488) bytes.
> summarize unreferenced
Unsigned allocations have 70 instances taking 0x20d0(8,400) bytes.
   Unsigned allocations of size 0x18 have 21 instances taking 0x1f8(504) bytes.
   Unsigned allocations of size 0x58 have 11 instances taking 0x3c8(968) bytes.
   Unsigned allocations of size 0x48 have 9 instances taking 0x288(648) bytes.
   Unsigned allocations of size 0x28 have 8 instances taking 0x140(320) bytes.
   Unsigned allocations of size 0x38 have 7 instances taking 0x188(392) bytes.
   Unsigned allocations of size 0x208 have 4 instances taking 0x820(2,080) bytes
.
   Unsigned allocations of size 0x88 have 3 instances taking 0x198(408) bytes.
   Unsigned allocations of size 0x308 have 3 instances taking 0x918(2,328) bytes
.
   Unsigned allocations of size 0x98 have 2 instances taking 0x130(304) bytes.
   Unsigned allocations of size 0xb8 have 1 instances taking 0xb8(184) bytes.
   Unsigned allocations of size 0x108 have 1 instances taking 0x108(264) bytes.
Signature 7fb5712cfb90 (XMP_Node) has 1 instances taking 0x58(88) bytes.
71 allocations use 0x2128 (8,488) bytes.
> 
```

Summarizing Unreferenced Allocations



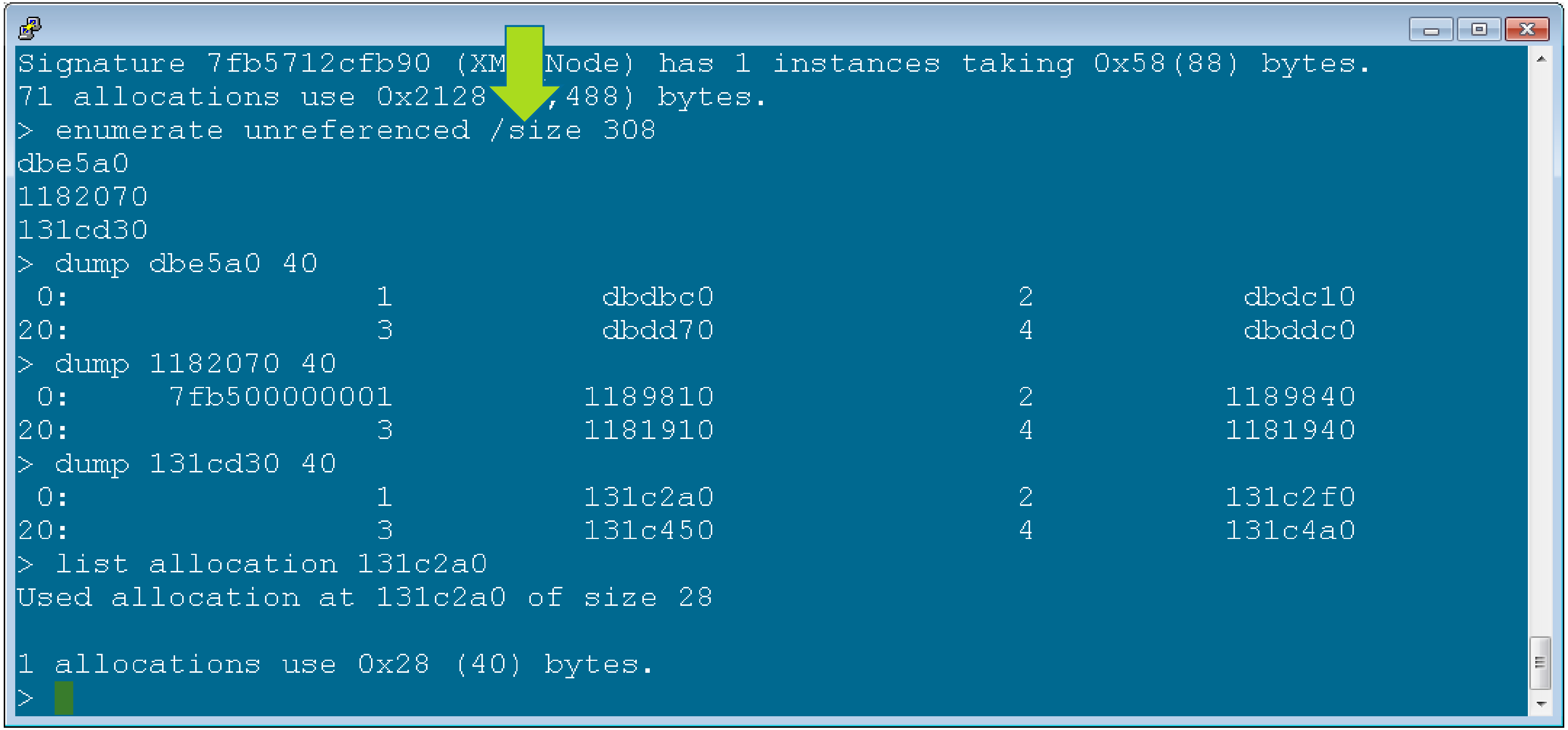
```
> count unreferenced
71 allocations use 0x2128 (8,488) bytes.
> summarize unreferenced
Unsigned allocations have 70 instances taking 0x20d0(8,400) bytes.
   Unsigned allocations of size 0x18 have 21 instances taking 0x1f8(504) bytes.
   Unsigned allocations of size 0x58 have 11 instances taking 0x3c8(968) bytes.
   Unsigned allocations of size 0x48 have 9 instances taking 0x288(648) bytes.
   Unsigned allocations of size 0x28 have 8 instances taking 0x140(320) bytes.
   Unsigned allocations of size 0x38 have 7 instances taking 0x188(392) bytes.
   Unsigned allocations of size 0x208 have 4 instances taking 0x820(2,080) bytes
.
   Unsigned allocations of size 0x88 have 3 instances taking 0x198(408) bytes.
   Unsigned allocations of size 0x308 have 3 instances taking 0x918(2,328) bytes
.
   Unsigned allocations of size 0x98 have 2 instances taking 0x130(304) bytes.
   Unsigned allocations of size 0xb8 have 1 instances taking 0xb8(184) bytes.
   Unsigned allocations of size 0x108 have 1 instances taking 0x108(264) bytes.
Signature 7fb5712cfb90 (XMP_Node) has 1 instances taking 0x58(88) bytes.
71 allocations use 0x2128 (8,488) bytes.
>
```

Looking at Similar Leaks



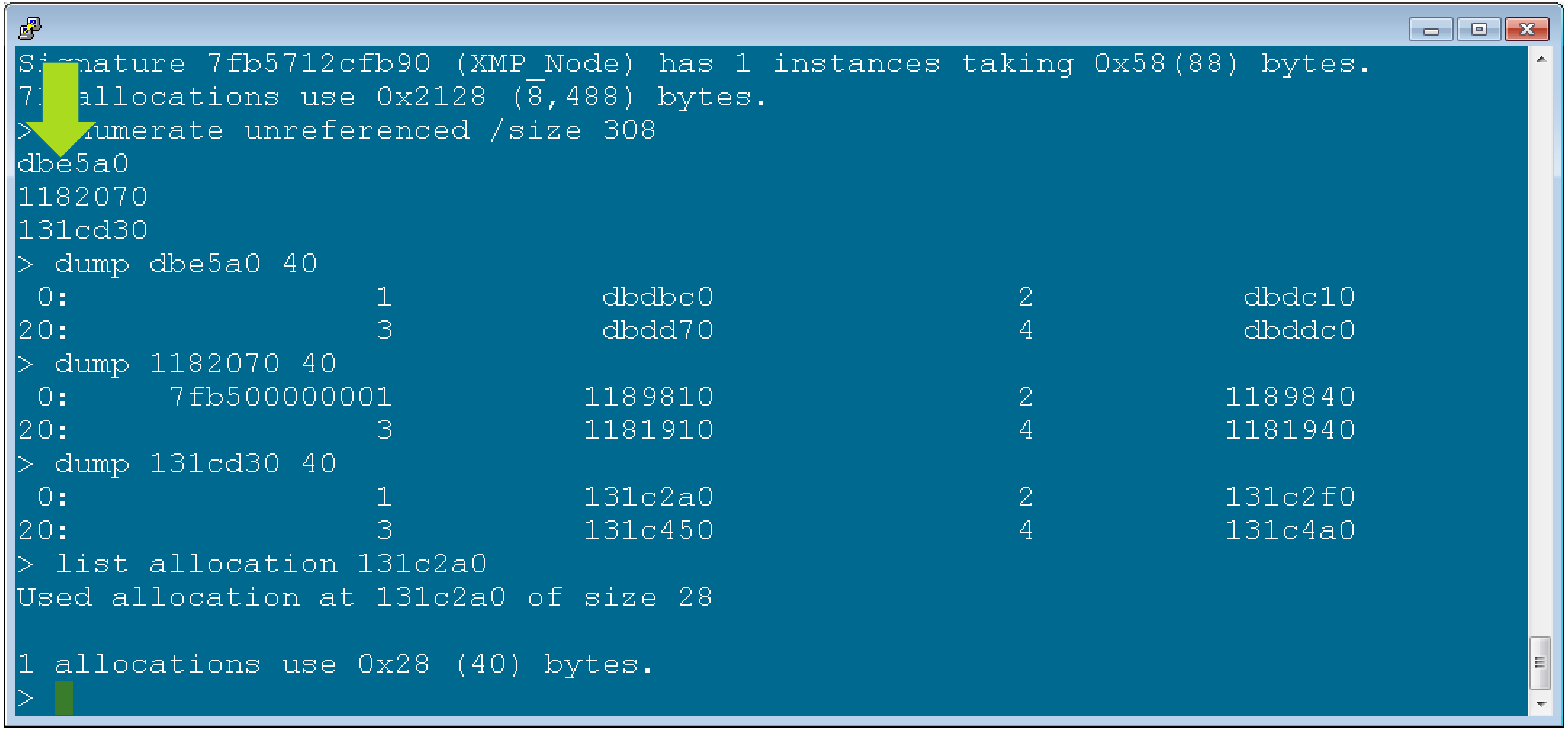
```
Signature 7fb5712cfb90 (XMP_Node) has 1 instances taking 0x58(88) bytes.
71 allocations use 0x2128 (8,488) bytes.
> enumerate unreferenced /size 308
dbe5a0
1182070
131cd30
> dump dbe5a0 40
 0:                1           dbdbc0                2           dbdc10
20:                3           dbdd70                4           dbddc0
> dump 1182070 40
 0:     7fb500000001          1189810                2          1189840
20:                3          1181910                4          1181940
> dump 131cd30 40
 0:                1          131c2a0                2          131c2f0
20:                3          131c450                4          131c4a0
> list allocation 131c2a0
Used allocation at 131c2a0 of size 28

1 allocations use 0x28 (40) bytes.
> 
```

Looking at Similar Leaks



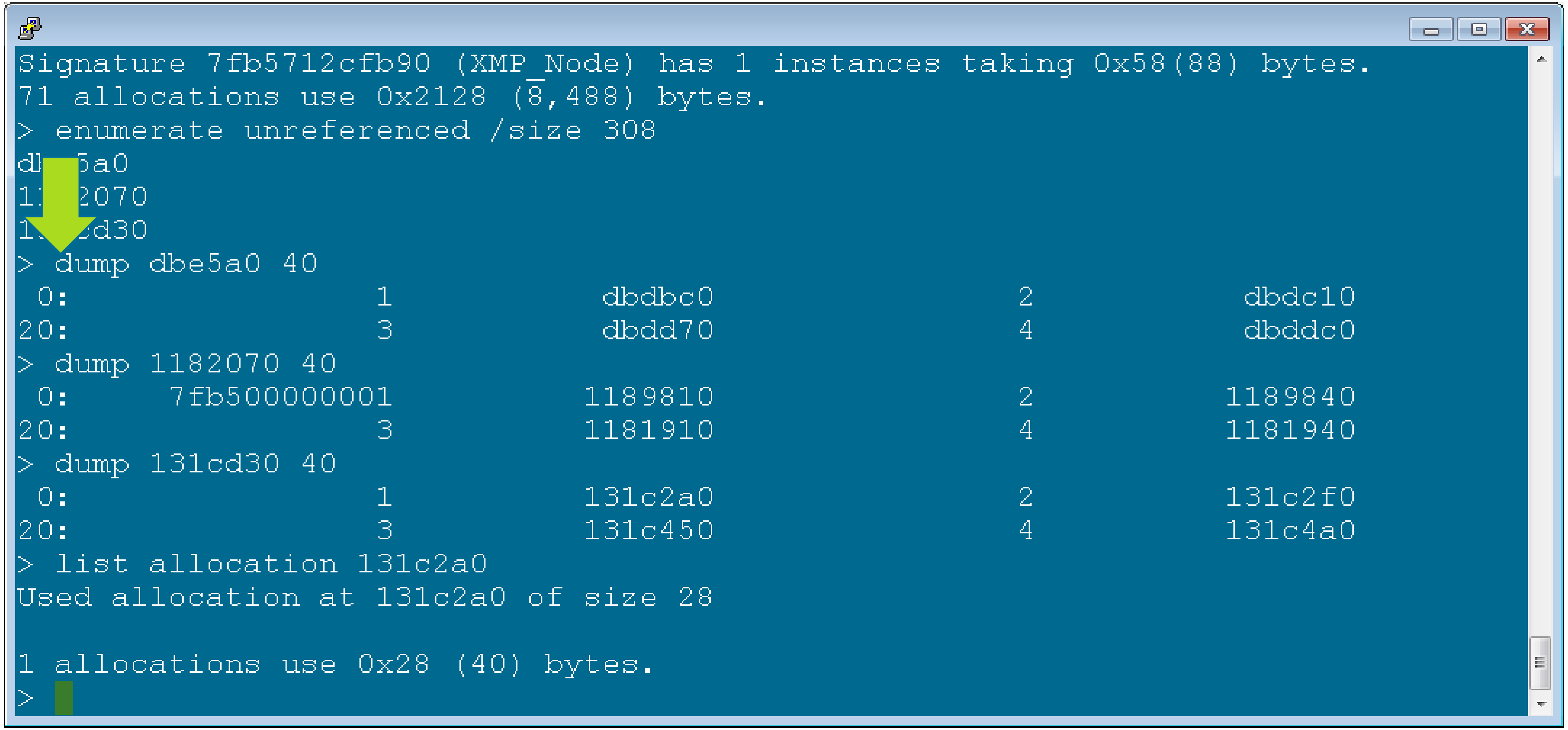
```
Signature 7fb5712cfb90 (XM  Node) has 1 instances taking 0x58(88) bytes.
71 allocations use 0x2128   ,488) bytes.
> enumerate unreferenced /size 308
dbe5a0
1182070
131cd30
> dump dbe5a0 40
 0:                1           dbdbc0                2           dbdc10
20:                3           dbdd70                4           dbddc0
> dump 1182070 40
 0:     7fb500000001          1189810                2          1189840
20:                3          1181910                4          1181940
> dump 131cd30 40
 0:                1          131c2a0                2          131c2f0
20:                3          131c450                4          131c4a0
> list allocation 131c2a0
Used allocation at 131c2a0 of size 28

1 allocations use 0x28 (40) bytes.
> 
```

Looking at Similar Leaks



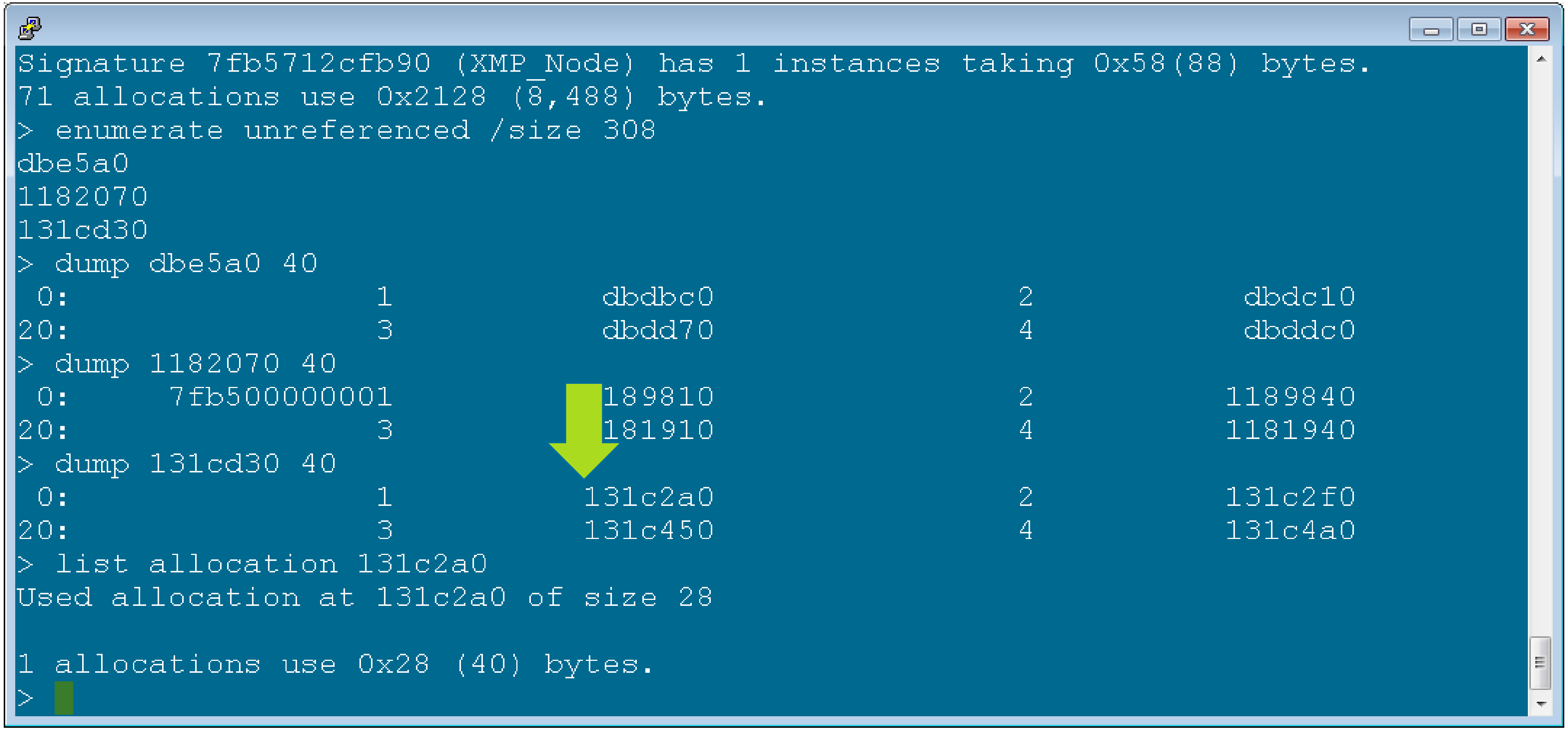
```
Signature 7fb5712cfb90 (XMP_Node) has 1 instances taking 0x58(88) bytes.
71 allocations use 0x2128 (8,488) bytes.
> enumerate unreferenced /size 308
dbe5a0
1182070
131cd30
> dump dbe5a0 40
 0:                  1           dbdbc0                  2            dbdc10
20:                  3           dbdd70                  4            dbddc0
> dump 1182070 40
 0:       7fb500000001          1189810                  2           1189840
20:                  3          1181910                  4           1181940
> dump 131cd30 40
 0:                  1          131c2a0                  2           131c2f0
20:                  3          131c450                  4           131c4a0
> list allocation 131c2a0
Used allocation at 131c2a0 of size 28

1 allocations use 0x28 (40) bytes.
>
```

Looking at Similar Leaks



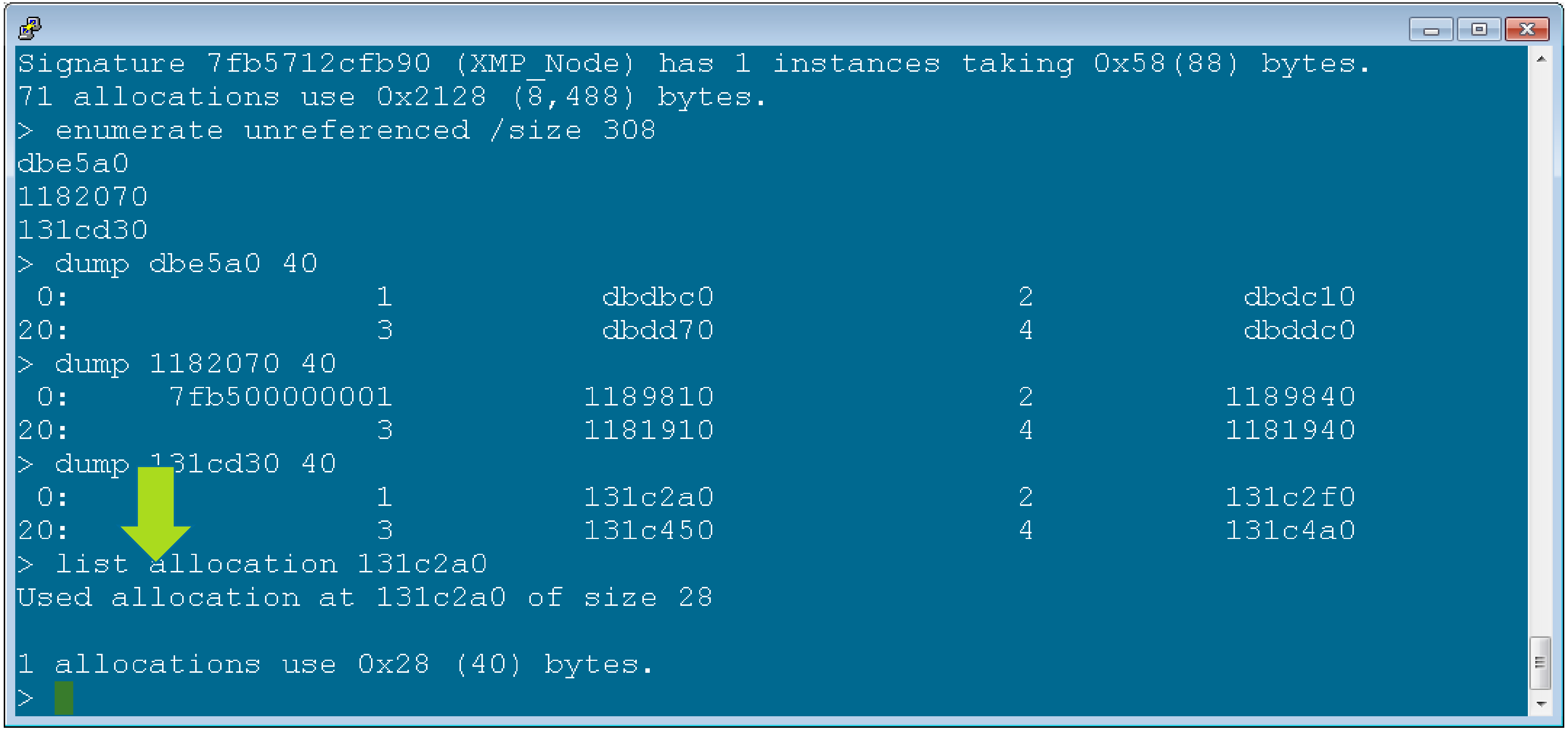
```
Signature 7fb5712cfb90 (XMP_Node) has 1 instances taking 0x58(88) bytes.
71 allocations use 0x2128 (8,488) bytes.
> enumerate unreferenced /size 308
dbe5a0
1182070
131cd30
> dump dbe5a0 40
 0:                1           dbdbc0                2           dbdc10
20:                3           dbdd70                4           dbddc0
> dump 1182070 40
 0:     7fb500000001          1189810                2          1189840
20:                3          1181910                4          1181940
> dump 131cd30 40
 0:                1          131c2a0                2          131c2f0
20:                3          131c450                4          131c4a0
> list allocation 131c2a0
Used allocation at 131c2a0 of size 28

1 allocations use 0x28 (40) bytes.
> 
```

Looking at Similar Leaks



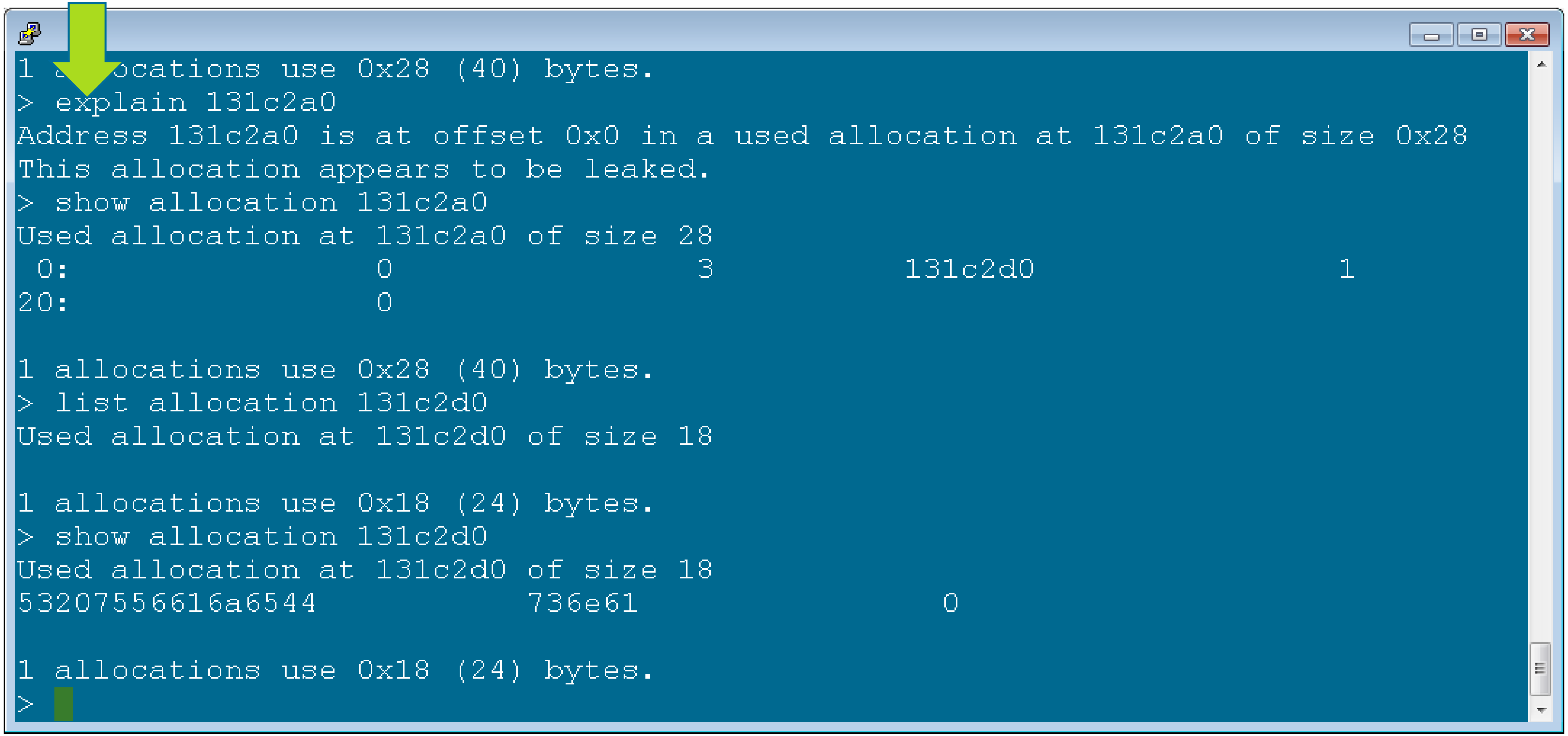
```
Signature 7fb5712cfb90 (XMP_Node) has 1 instances taking 0x58(88) bytes.
71 allocations use 0x2128 (8,488) bytes.
> enumerate unreferenced /size 308
dbe5a0
1182070
131cd30
> dump dbe5a0 40
 0:                1           dbdbc0                1           dbdc10
20:                3           dbdd70                4           dbddc0
> dump 1182070 40
 0:       7fb500000001          189810                2          1189840
20:                3            181910                4          1181940
> dump 131cd30 40
 0:                1          131c2a0                2          131c2f0
20:                3          131c450                4          131c4a0
> list allocation 131c2a0
Used allocation at 131c2a0 of size 28

1 allocations use 0x28 (40) bytes.
>
```

Looking at Similar Leaks



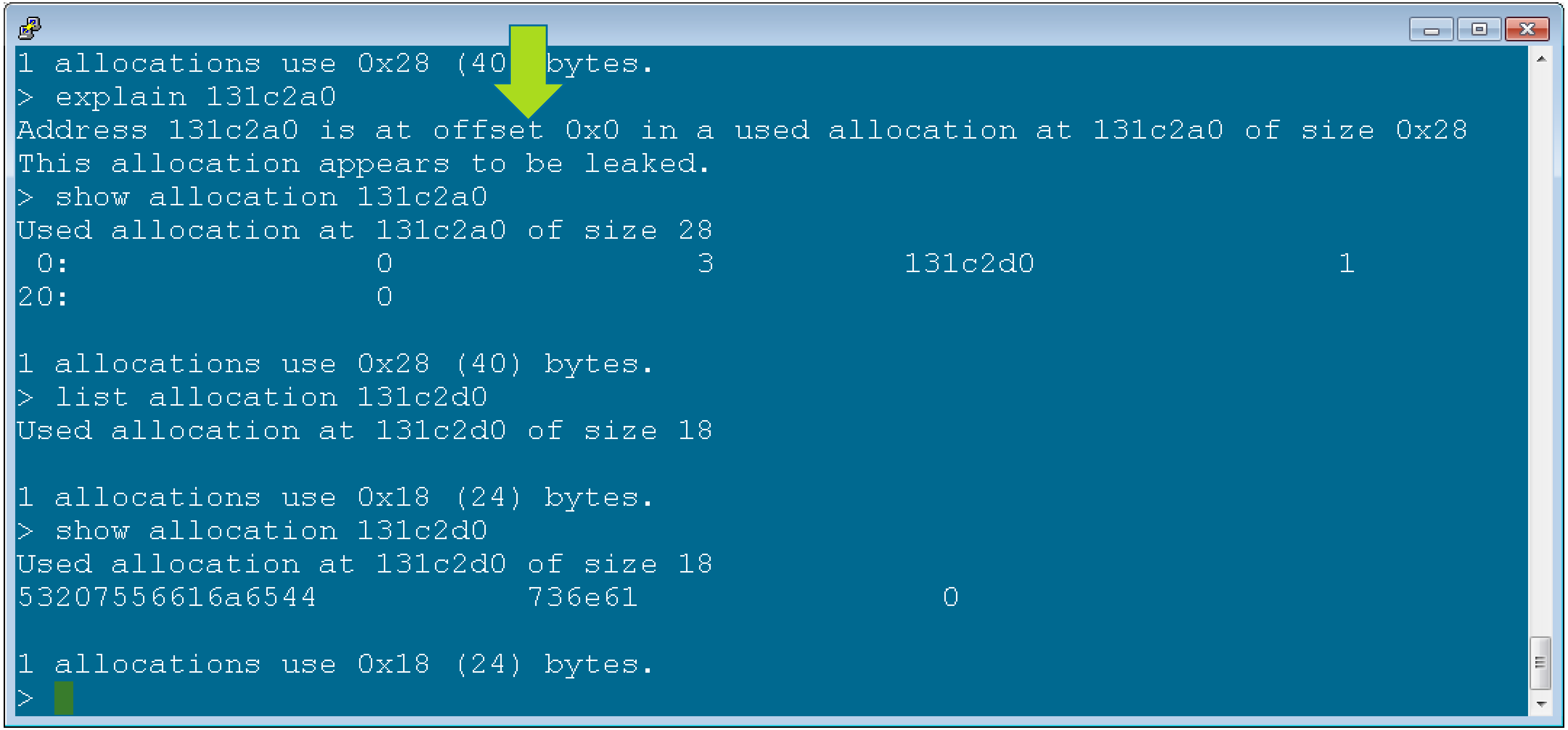
```
Signature 7fb5712cfb90 (XMP_Node) has 1 instances taking 0x58(88) bytes.
71 allocations use 0x2128 (8,488) bytes.
> enumerate unreferenced /size 308
dbe5a0
1182070
131cd30
> dump dbe5a0 40
 0:                 1           dbdbc0                   2              dbdc10
20:                 3           dbdd70                   4              dbddc0
> dump 1182070 40
 0:       7fb500000001         1189810                   2             1189840
20:                 3          1181910                   4             1181940
> dump 131cd30 40
 0:                 1          131c2a0                   2             131c2f0
20:                 3          131c450                   4             131c4a0
> list allocation 131c2a0
Used allocation at 131c2a0 of size 28

1 allocations use 0x28 (40) bytes.
> 
```

Following Outgoing Edges



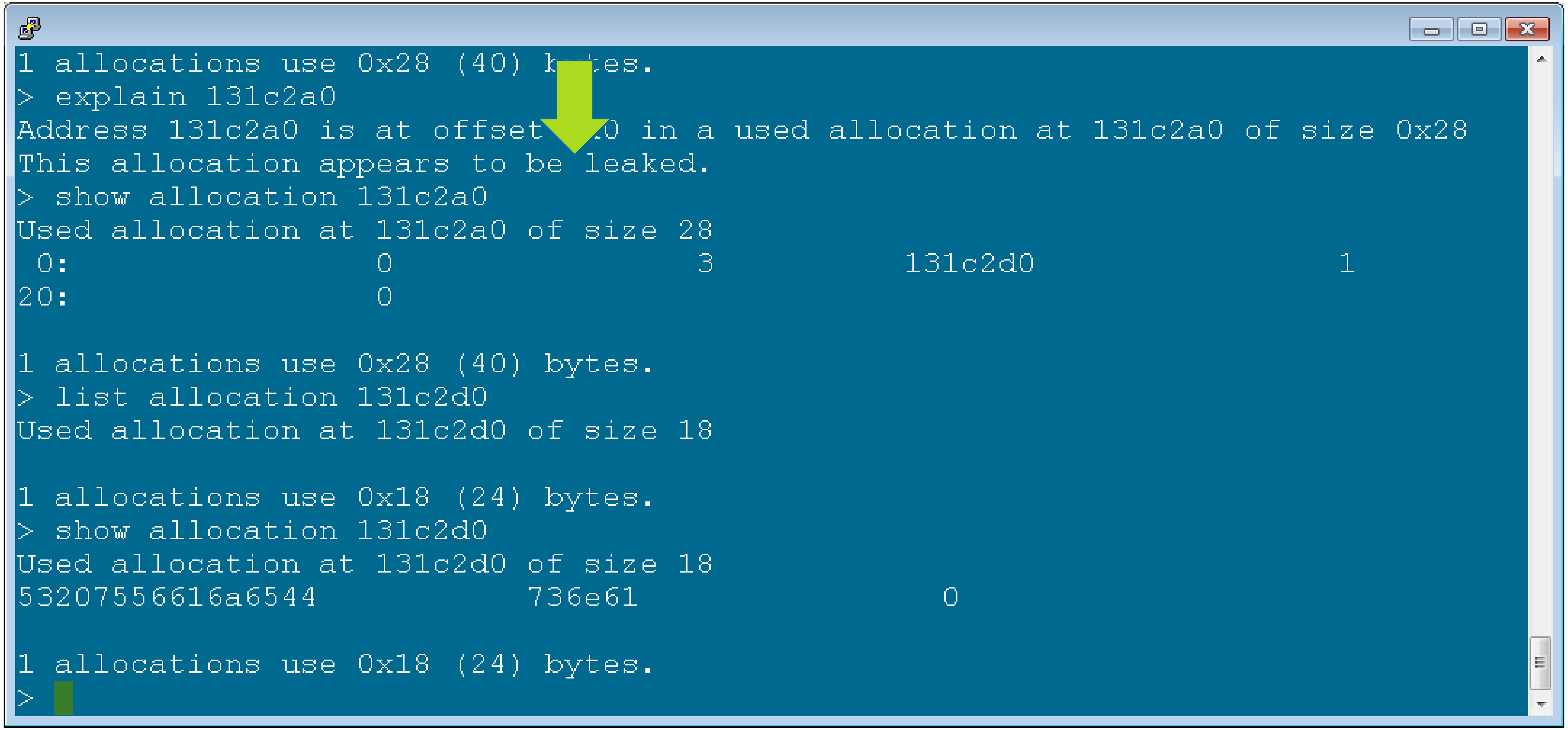
```
1 allocations use 0x28 (40) bytes.
> explain 131c2a0
Address 131c2a0 is at offset 0x0 in a used allocation at 131c2a0 of size 0x28
This allocation appears to be leaked.
> show allocation 131c2a0
Used allocation at 131c2a0 of size 28
 0:                0                3          131c2d0                 1
20:                0

1 allocations use 0x28 (40) bytes.
> list allocation 131c2d0
Used allocation at 131c2d0 of size 18

1 allocations use 0x18 (24) bytes.
> show allocation 131c2d0
Used allocation at 131c2d0 of size 18
5320755616a6544            736e61                0

1 allocations use 0x18 (24) bytes.
> 
```

Following Outgoing Edges



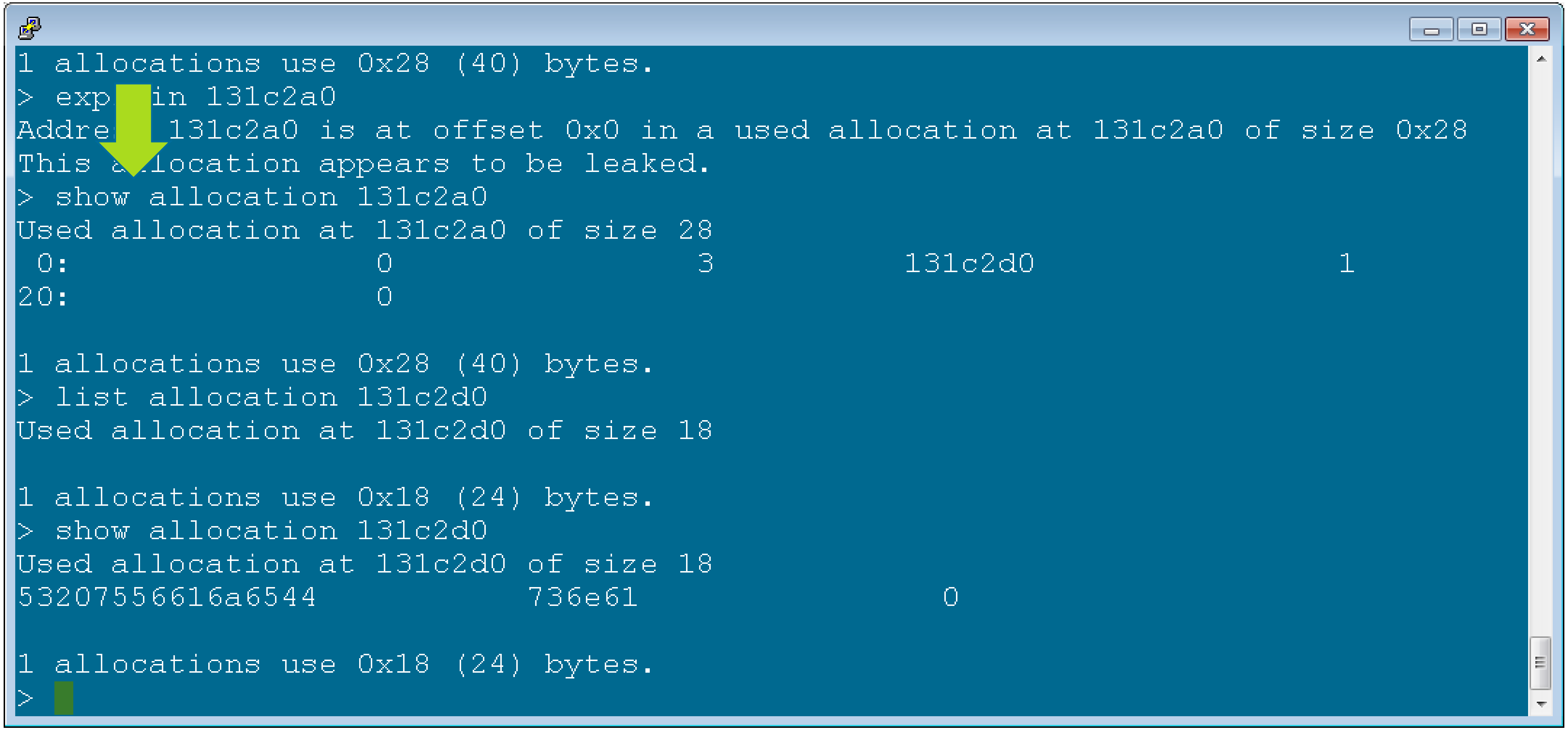
```
1 allocations use 0x28 (40 bytes.
> explain 131c2a0
Address 131c2a0 is at offset 0x0 in a used allocation at 131c2a0 of size 0x28
This allocation appears to be leaked.
> show allocation 131c2a0
Used allocation at 131c2a0 of size 28
 0:                 0                3          131c2d0                1
20:                 0

1 allocations use 0x28 (40) bytes.
> list allocation 131c2d0
Used allocation at 131c2d0 of size 18

1 allocations use 0x18 (24) bytes.
> show allocation 131c2d0
Used allocation at 131c2d0 of size 18
53207556616a6544           736e61                     0

1 allocations use 0x18 (24) bytes.
>
```

Following Outgoing Edges



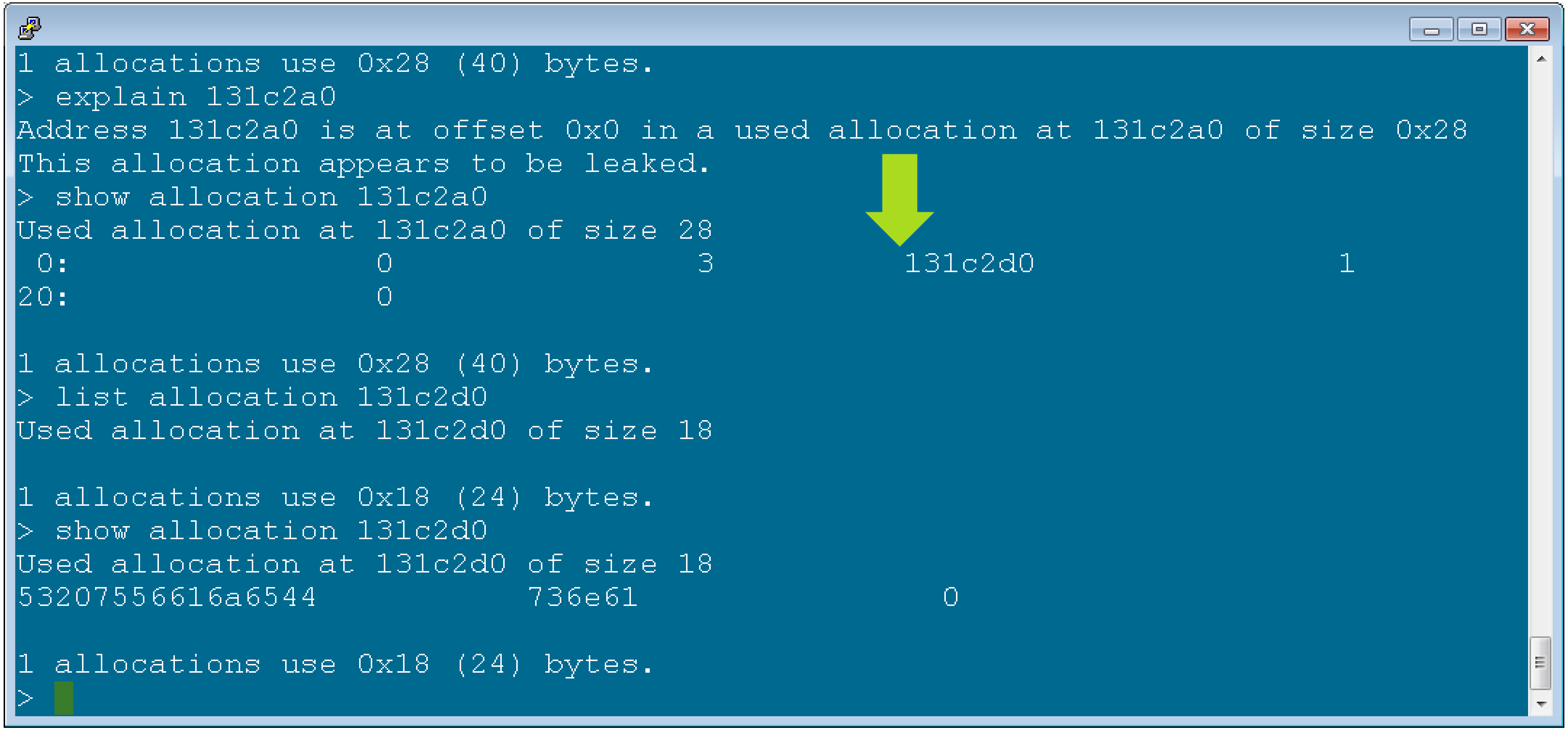
```
1 allocations use 0x28 (40) bytes.
> explain 131c2a0
Address 131c2a0 is at offset 0 in a used allocation at 131c2a0 of size 0x28
This allocation appears to be leaked.
> show allocation 131c2a0
Used allocation at 131c2a0 of size 28
 0:                0                3          131c2d0                1
20:                0

1 allocations use 0x28 (40) bytes.
> list allocation 131c2d0
Used allocation at 131c2d0 of size 18

1 allocations use 0x18 (24) bytes.
> show allocation 131c2d0
Used allocation at 131c2d0 of size 18
53207556616a6544           736e61                0

1 allocations use 0x18 (24) bytes.
>
```

Following Outgoing Edges



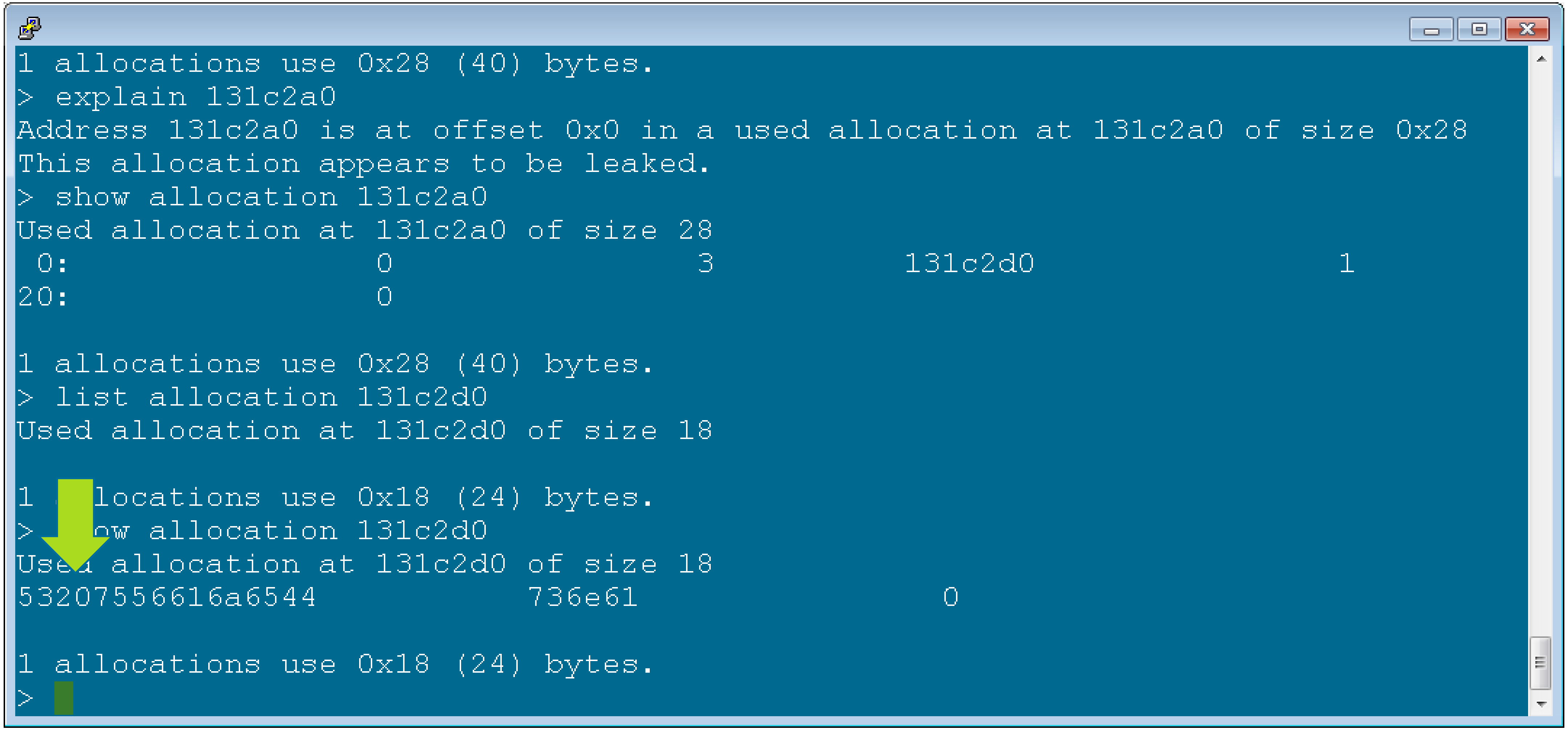
```
1 allocations use 0x28 (40) bytes.
> explain 131c2a0
Address 131c2a0 is at offset 0x0 in a used allocation at 131c2a0 of size 0x28
This allocation appears to be leaked.
> show allocation 131c2a0
Used allocation at 131c2a0 of size 28
 0:                0                3          131c2d0                1
20:                0

1 allocations use 0x28 (40) bytes.
> list allocation 131c2d0
Used allocation at 131c2d0 of size 18

1 allocations use 0x18 (24) bytes.
> show allocation 131c2d0
Used allocation at 131c2d0 of size 18
53207556616a6544           736e61                0

1 allocations use 0x18 (24) bytes.
>
```

Following Outgoing Edges



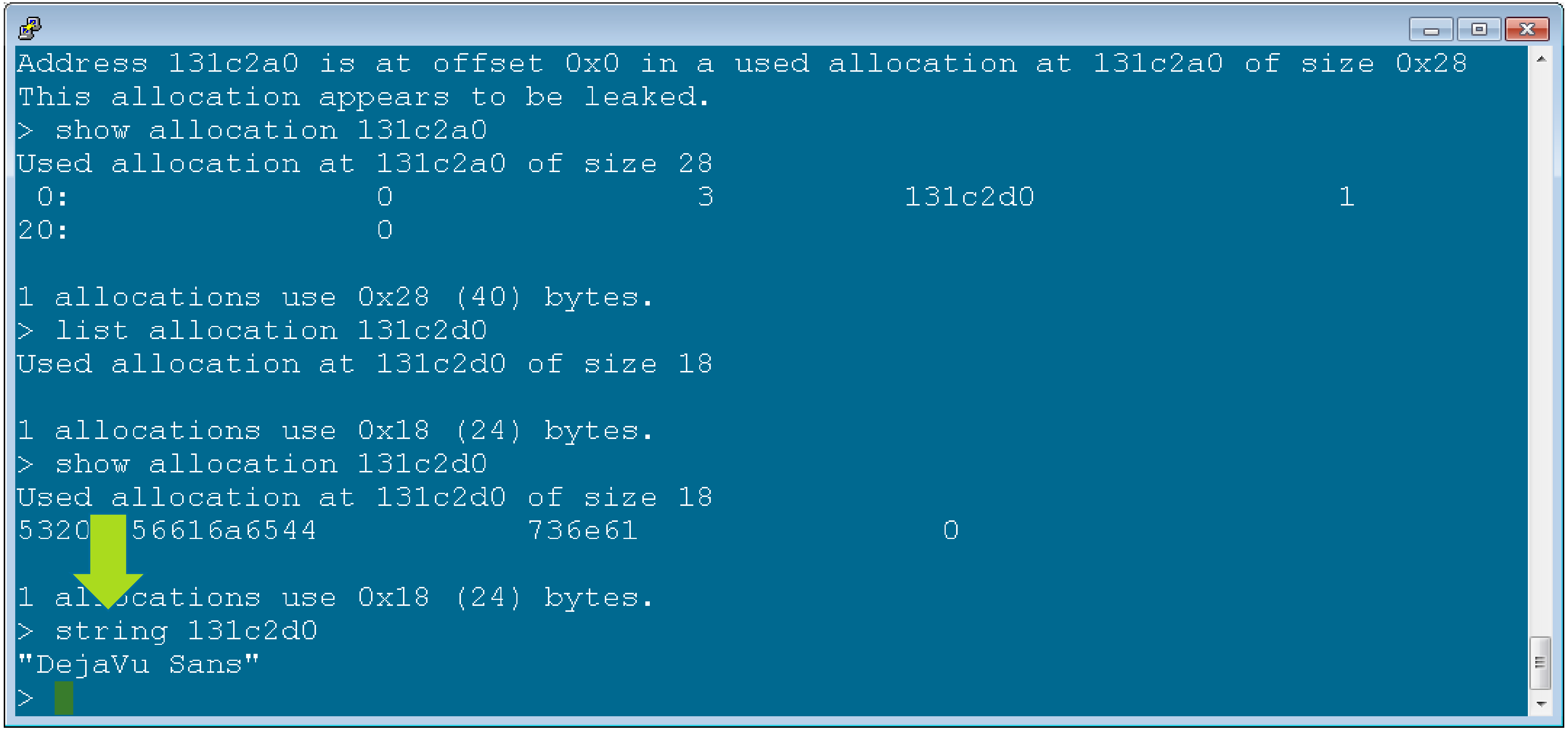
```
1 allocations use 0x28 (40) bytes.
> explain 131c2a0
Address 131c2a0 is at offset 0x0 in a used allocation at 131c2a0 of size 0x28
This allocation appears to be leaked.
> show allocation 131c2a0
Used allocation at 131c2a0 of size 28
 0:                0                3          131c2d0                1
20:                0

1 allocations use 0x28 (40) bytes.
> list allocation 131c2d0
Used allocation at 131c2d0 of size 18

1 allocations use 0x18 (24) bytes.
> show allocation 131c2d0
Used allocation at 131c2d0 of size 18
53207556616a6544           736e61                0

1 allocations use 0x18 (24) bytes.
> 
```

Following Outgoing Edges



```
1 allocations use 0x28 (40) bytes.
> explain 131c2a0
Address 131c2a0 is at offset 0x0 in a used allocation at 131c2a0 of size 0x28
This allocation appears to be leaked.
> show allocation 131c2a0
Used allocation at 131c2a0 of size 28
 0:                0                3          131c2d0                1
20:                0

1 allocations use 0x28 (40) bytes.
> list allocation 131c2d0
Used allocation at 131c2d0 of size 18

1 allocations use 0x18 (24) bytes.
> show allocation 131c2d0
Used allocation at 131c2d0 of size 18
5320755616a6544            736e61                0

1 allocations use 0x18 (24) bytes.
> 
```

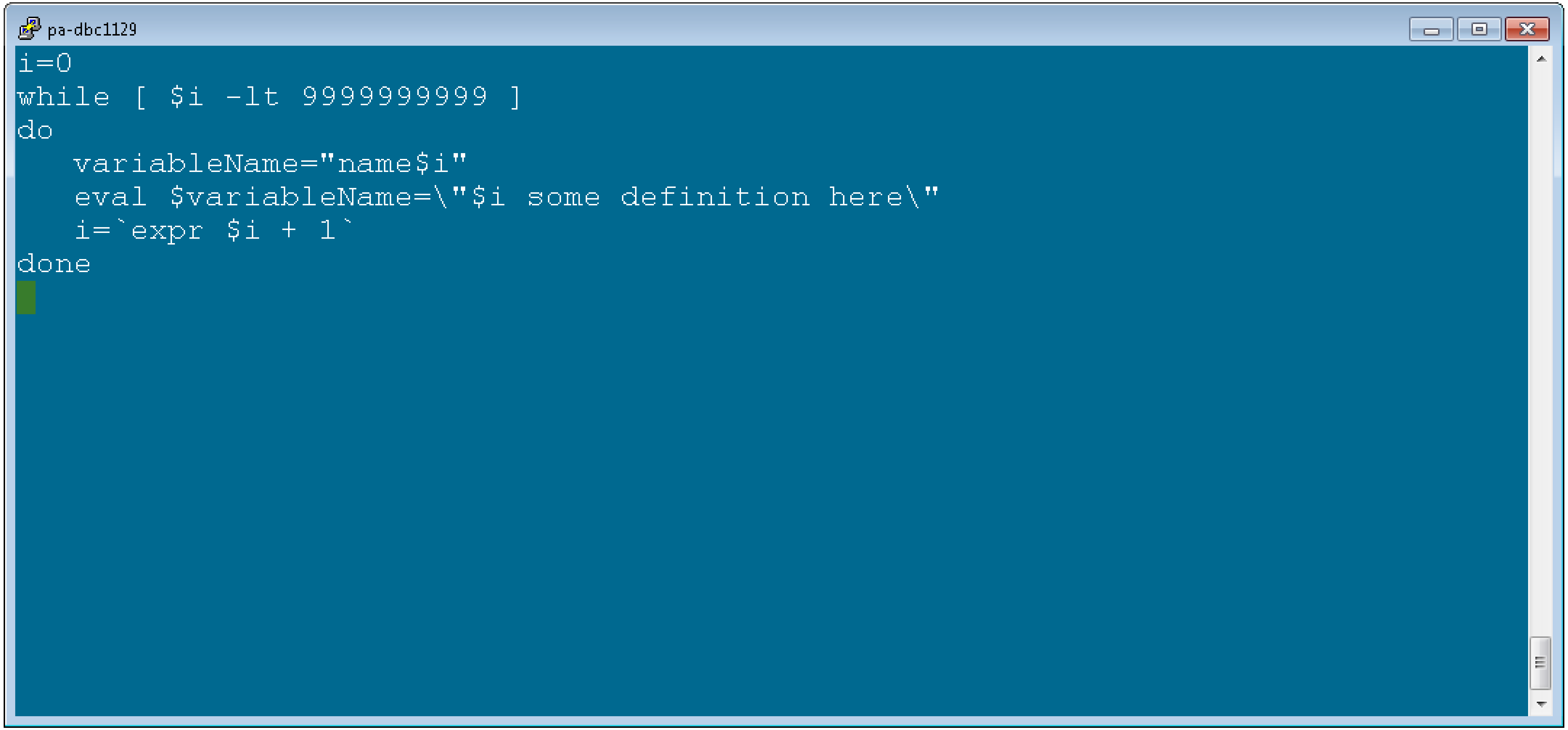
Detect and Analyze Memory Leaks – Looking at a String



```
Address 131c2a0 is at offset 0x0 in a used allocation at 131c2a0 of size 0x28
This allocation appears to be leaked.
> show allocation 131c2a0
Used allocation at 131c2a0 of size 28
 0:                0                3          131c2d0                1
20:                0

1 allocations use 0x28 (40) bytes.
> list allocation 131c2d0
Used allocation at 131c2d0 of size 18

1 allocations use 0x18 (24) bytes.
> show allocation 131c2d0
Used allocation at 131c2d0 of size 18
5320  56616a6544           736e61                0

1 allocations use 0x18 (24) bytes.
> string 131c2d0
"DejaVu Sans"
> 
```

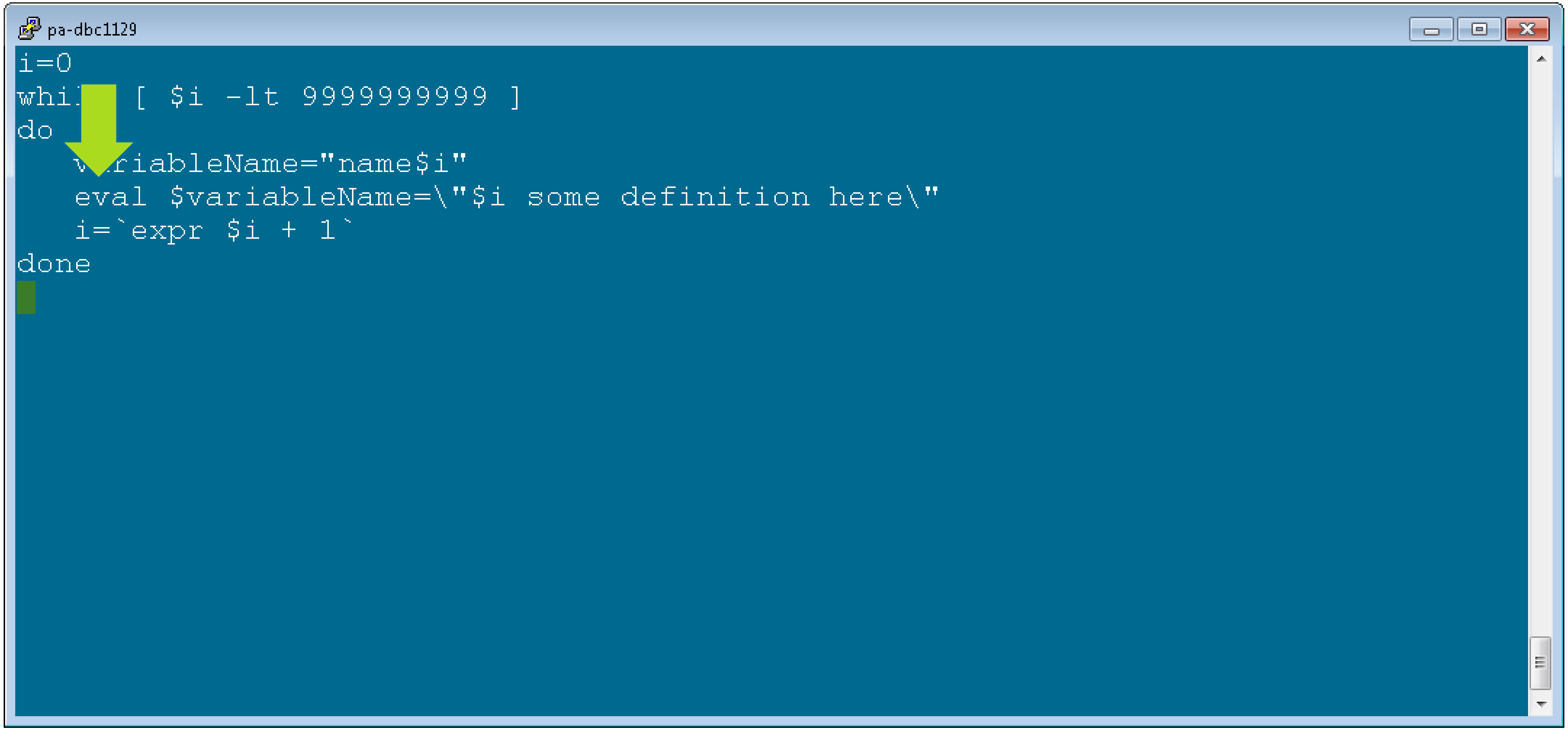
Using CHAP to Analyze Memory Growth

Analyzing Memory Growth



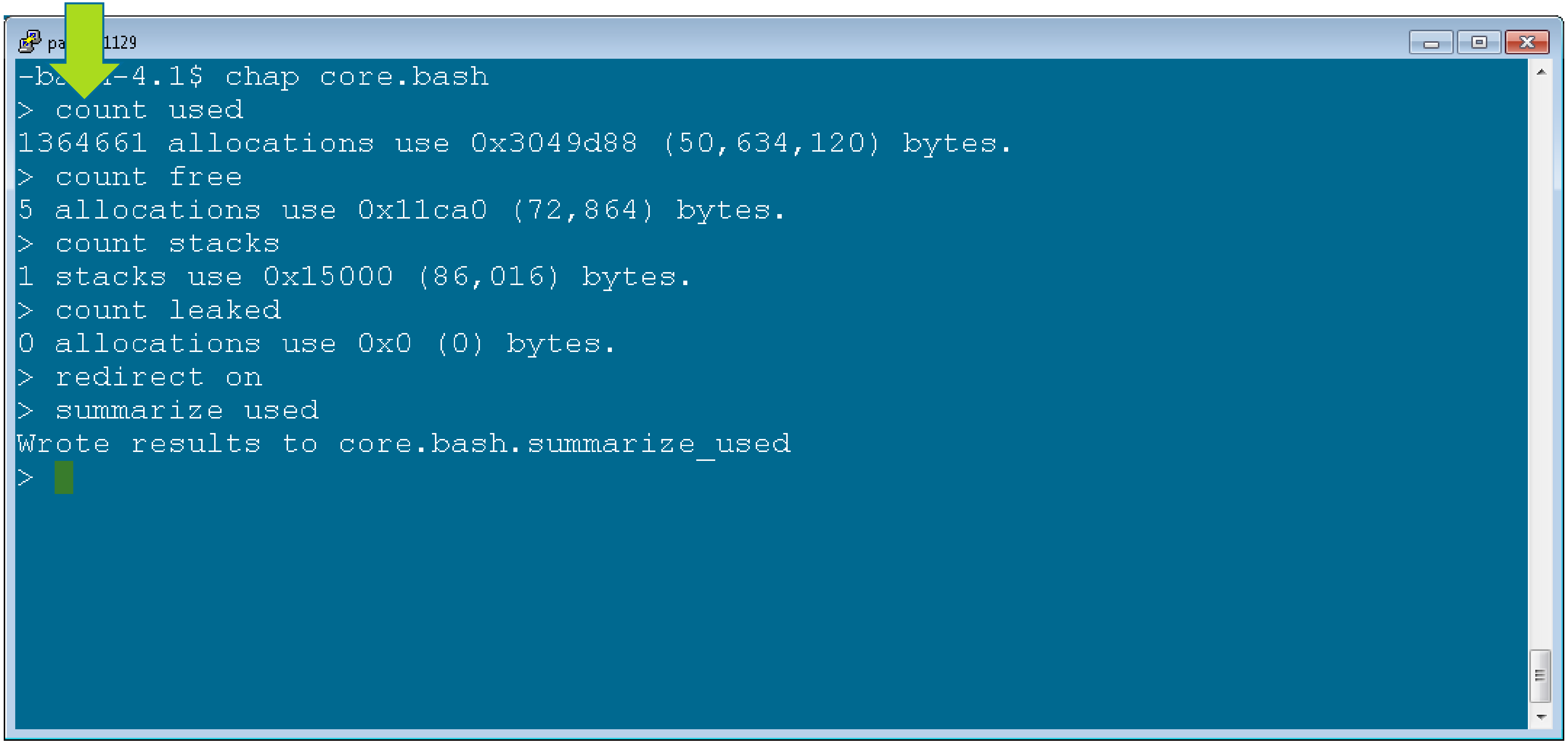
```
i=0
while [ $i -lt 9999999999 ]
do
   variableName="name$i"
   eval $variableName=\"$i some definition here\"
   i=`expr $i + 1`
done
```

Analyzing Memory Growth



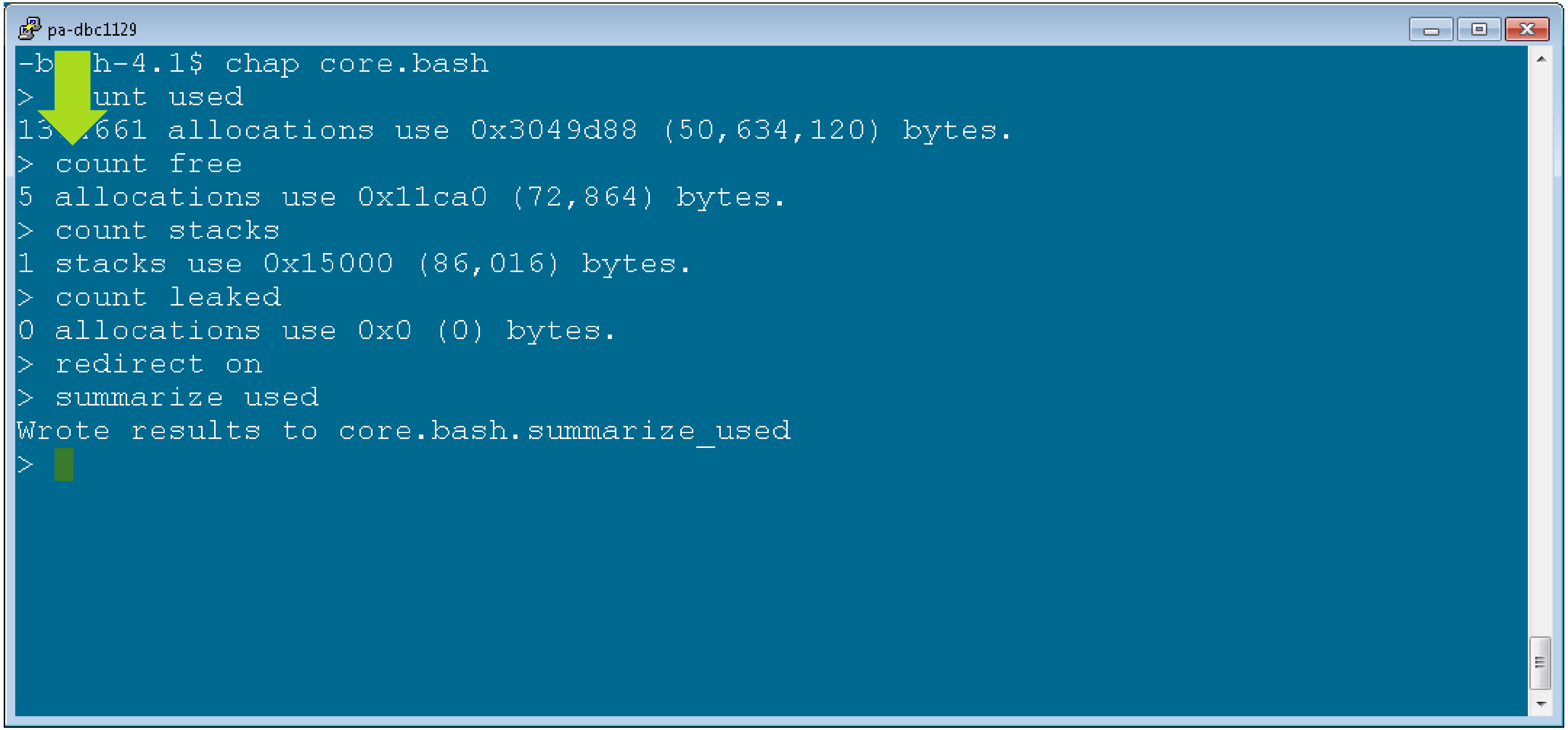
```
i=0
while [ $i -lt 9999999999 ]
do
   variableName="name$i"
   eval $variableName=\"$i some definition here\"
   i=`expr $i + 1`
done
```

Getting an Overview



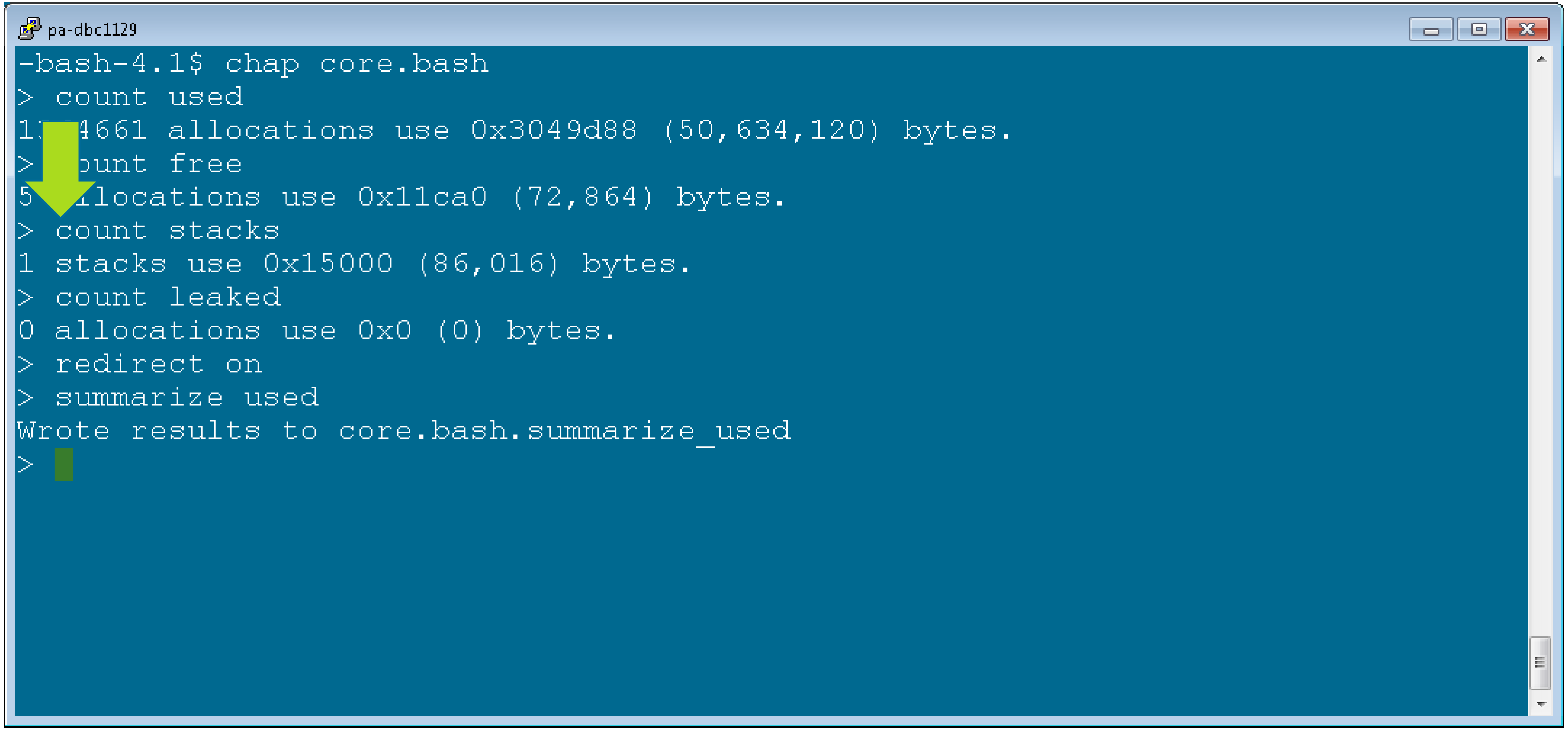
```
-bash-4.1$ chap core.bash
> count used
1364661 allocations use 0x3049d88 (50,634,120) bytes.
> count free
5 allocations use 0x11ca0 (72,864) bytes.
> count stacks
1 stacks use 0x15000 (86,016) bytes.
> count leaked
0 allocations use 0x0 (0) bytes.
> redirect on
> summarize used
Wrote results to core.bash.summarize_used
>
```

Getting an Overview



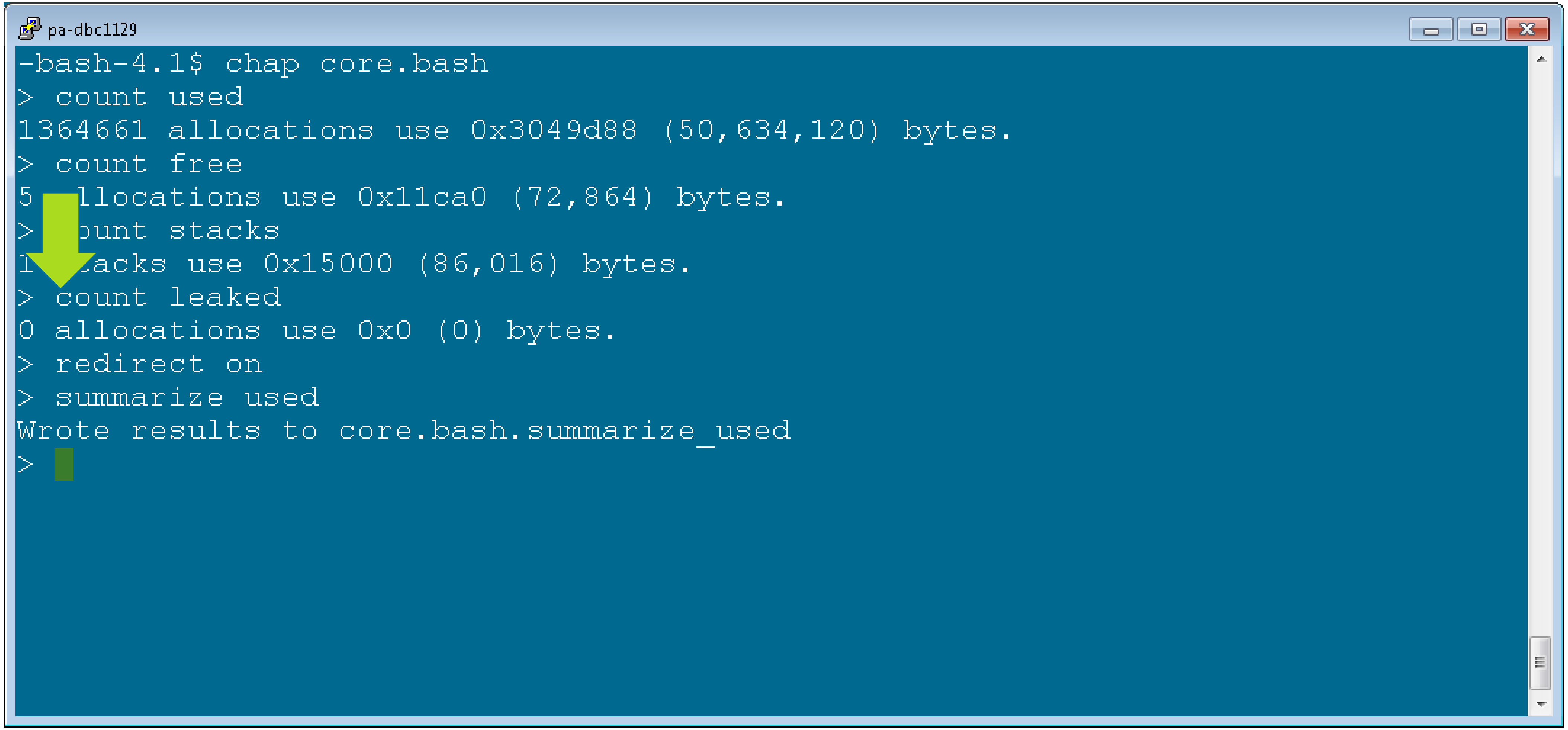
```
-bash-4.1$ chap core.bash
> count used
13[illegible]661 allocations use 0x3049d88 (50,634,120) bytes.
> count free
5 allocations use 0x11ca0 (72,864) bytes.
> count stacks
1 stacks use 0x15000 (86,016) bytes.
> count leaked
0 allocations use 0x0 (0) bytes.
> redirect on
> summarize used
Wrote results to core.bash.summarize_used
>
```

Getting an Overview



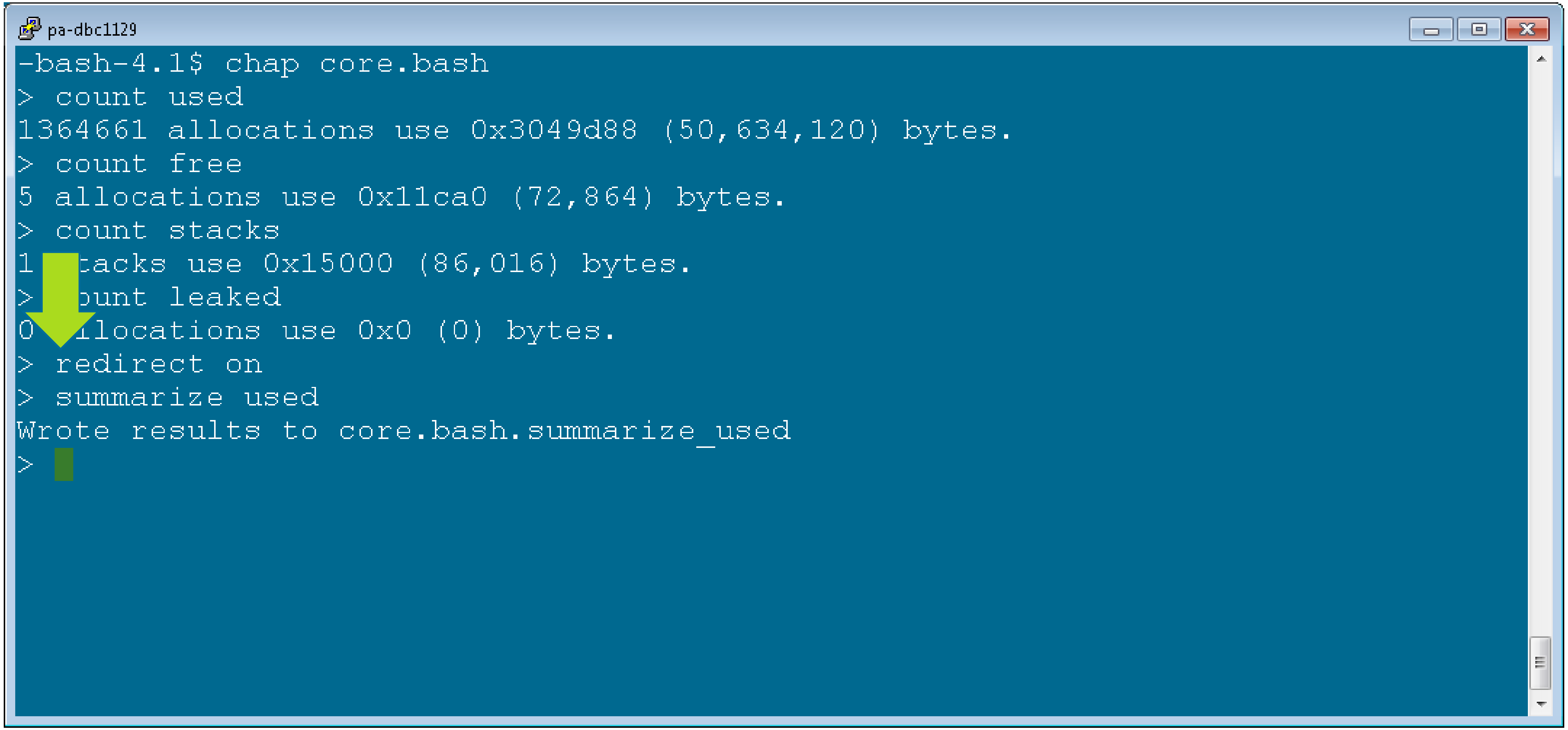
```
-bash-4.1$ chap core.bash
> count used
1[illegible]4661 allocations use 0x3049d88 (50,634,120) bytes.
> count free
5 allocations use 0x11ca0 (72,864) bytes.
> count stacks
1 stacks use 0x15000 (86,016) bytes.
> count leaked
0 allocations use 0x0 (0) bytes.
> redirect on
> summarize used
Wrote results to core.bash.summarize_used
> 
```

Getting an Overview



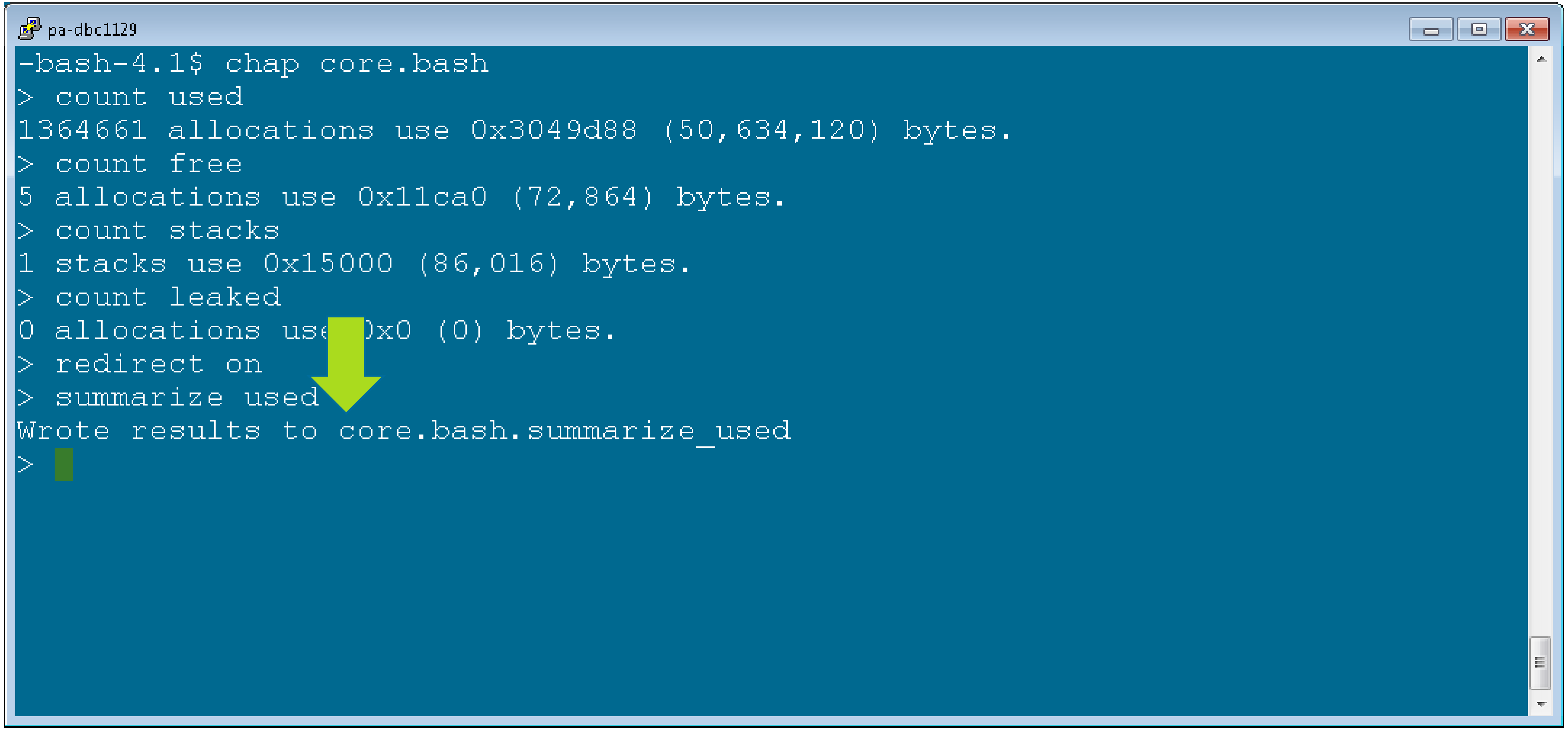
```
-bash-4.1$ chap core.bash
> count used
1364661 allocations use 0x3049d88 (50,634,120) bytes.
> count free
5 allocations use 0x11ca0 (72,864) bytes.
> count stacks
1 stacks use 0x15000 (86,016) bytes.
> count leaked
0 allocations use 0x0 (0) bytes.
> redirect on
> summarize used
Wrote results to core.bash.summarize_used
> 
```

Getting an Overview



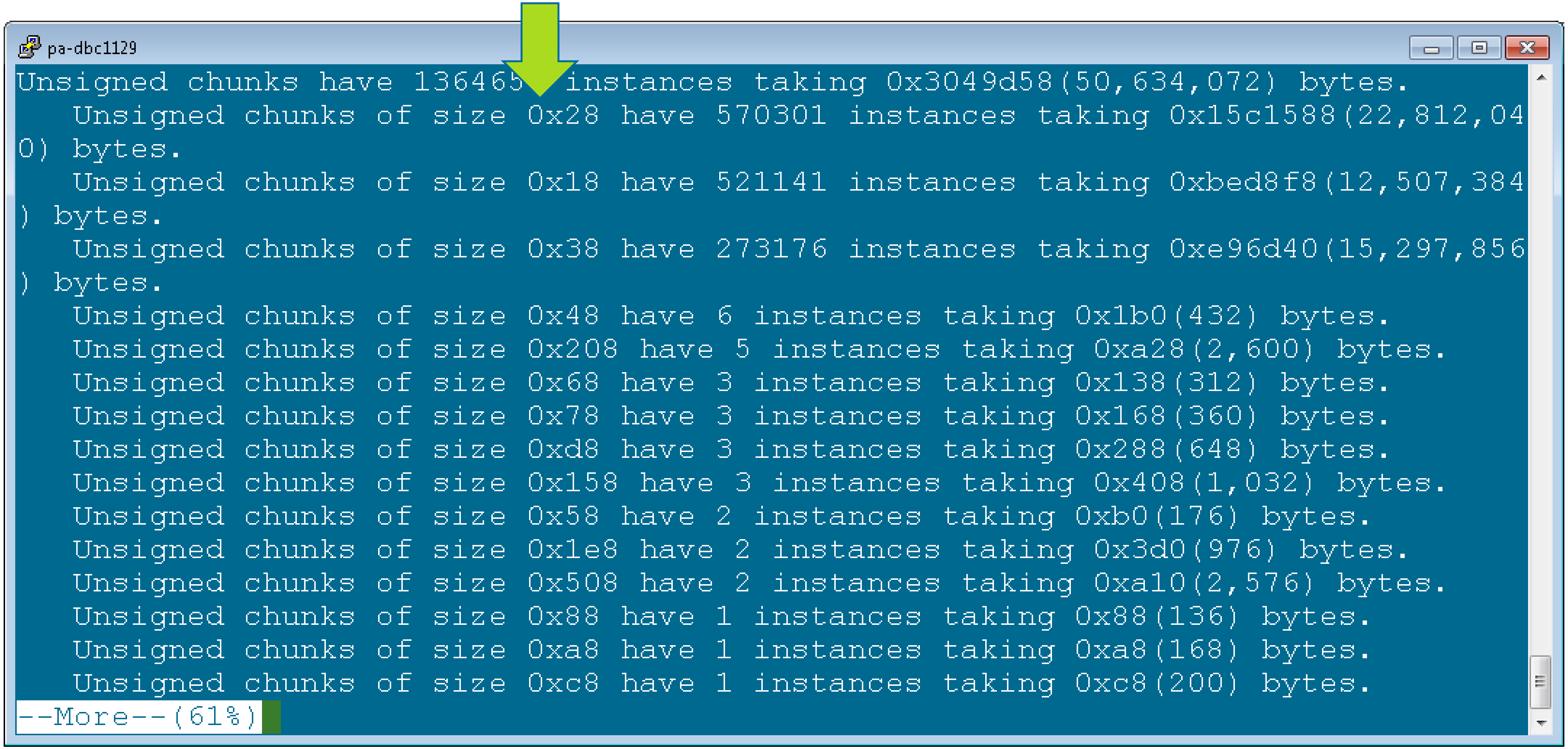
```
-bash-4.1$ chap core.bash
> count used
1364661 allocations use 0x3049d88 (50,634,120) bytes.
> count free
5 allocations use 0x11ca0 (72,864) bytes.
> count stacks
1 stacks use 0x15000 (86,016) bytes.
> count leaked
0 allocations use 0x0 (0) bytes.
> redirect on
> summarize used
Wrote results to core.bash.summarize_used
> 
```

Getting an Overview



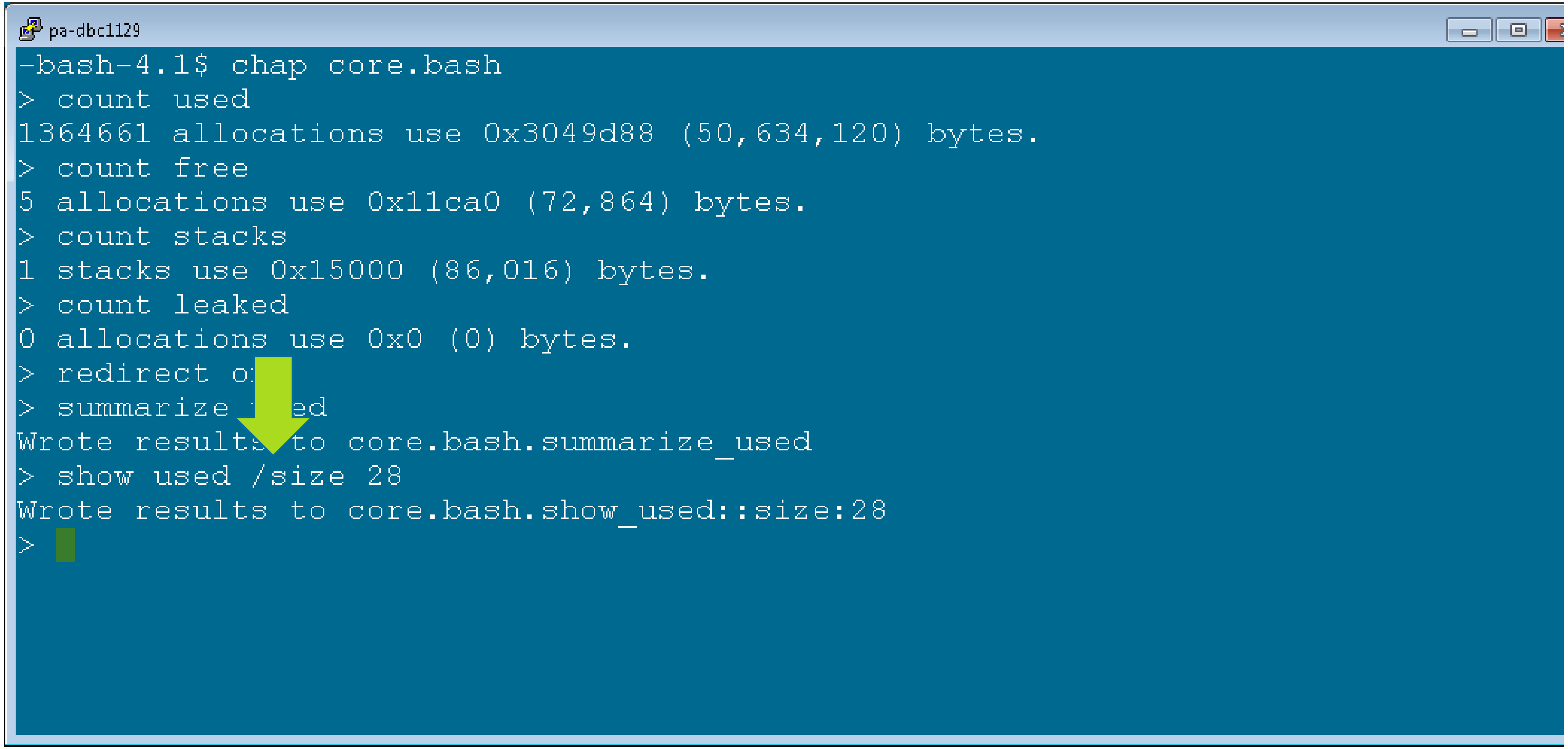
```
-bash-4.1$ chap core.bash
> count used
1364661 allocations use 0x3049d88 (50,634,120) bytes.
> count free
5 allocations use 0x11ca0 (72,864) bytes.
> count stacks
1 stacks use 0x15000 (86,016) bytes.
> count leaked
0 allocations use 0x0 (0) bytes.
> redirect on
> summarize used
Wrote results to core.bash.summarize_used
> 
```

Results of “summarize used”



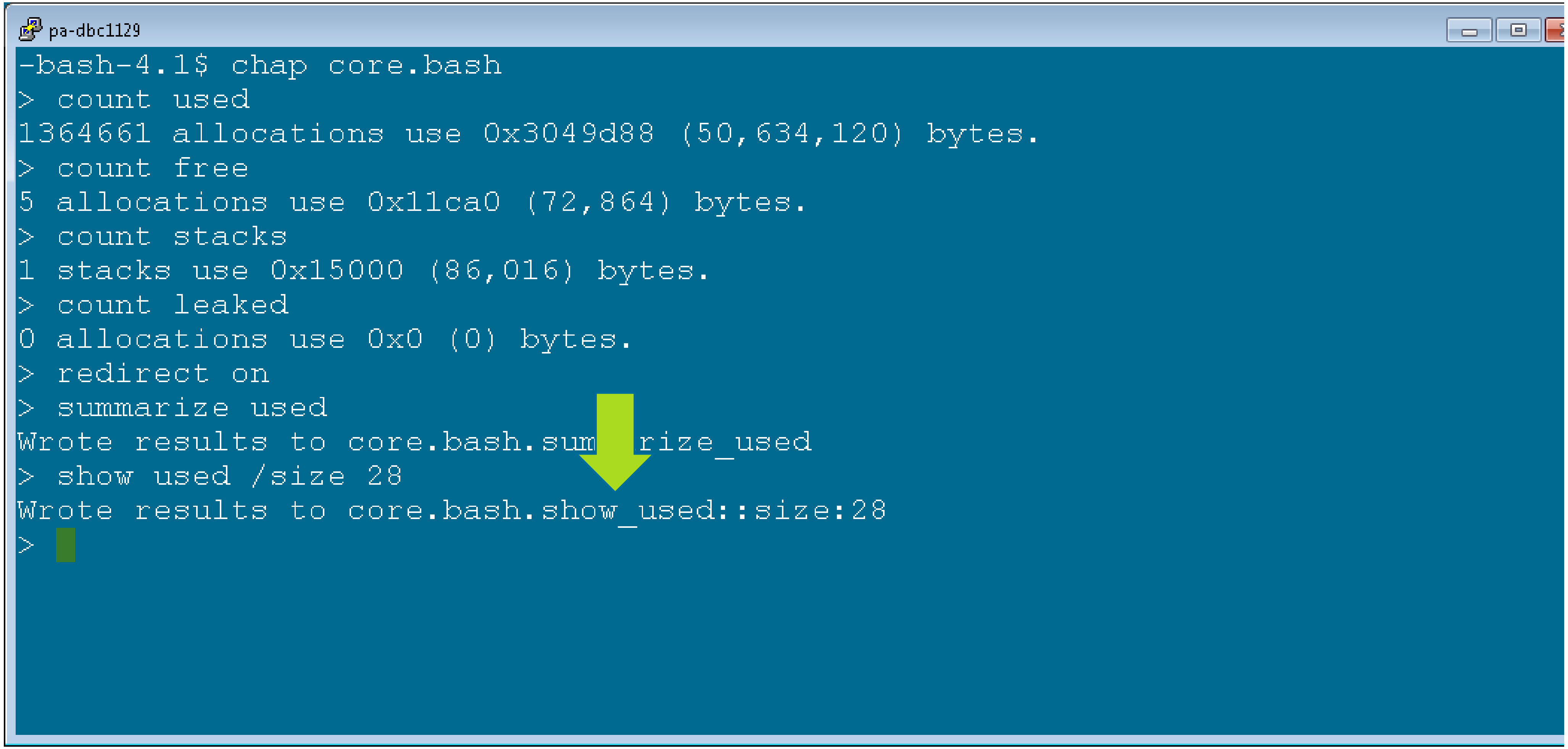
```
Unsigned chunks have 136465  instances taking 0x3049d58(50,634,072) bytes.
   Unsigned chunks of size 0x28 have 570301 instances taking 0x15c1588(22,812,04
0) bytes.
   Unsigned chunks of size 0x18 have 521141 instances taking 0xbed8f8(12,507,384
) bytes.
   Unsigned chunks of size 0x38 have 273176 instances taking 0xe96d40(15,297,856
) bytes.
   Unsigned chunks of size 0x48 have 6 instances taking 0x1b0(432) bytes.
   Unsigned chunks of size 0x208 have 5 instances taking 0xa28(2,600) bytes.
   Unsigned chunks of size 0x68 have 3 instances taking 0x138(312) bytes.
   Unsigned chunks of size 0x78 have 3 instances taking 0x168(360) bytes.
   Unsigned chunks of size 0xd8 have 3 instances taking 0x288(648) bytes.
   Unsigned chunks of size 0x158 have 3 instances taking 0x408(1,032) bytes.
   Unsigned chunks of size 0x58 have 2 instances taking 0xb0(176) bytes.
   Unsigned chunks of size 0x1e8 have 2 instances taking 0x3d0(976) bytes.
   Unsigned chunks of size 0x508 have 2 instances taking 0xa10(2,576) bytes.
   Unsigned chunks of size 0x88 have 1 instances taking 0x88(136) bytes.
   Unsigned chunks of size 0xa8 have 1 instances taking 0xa8(168) bytes.
   Unsigned chunks of size 0xc8 have 1 instances taking 0xc8(200) bytes.
--More--(61%)
```

Showing Many Allocations to a File



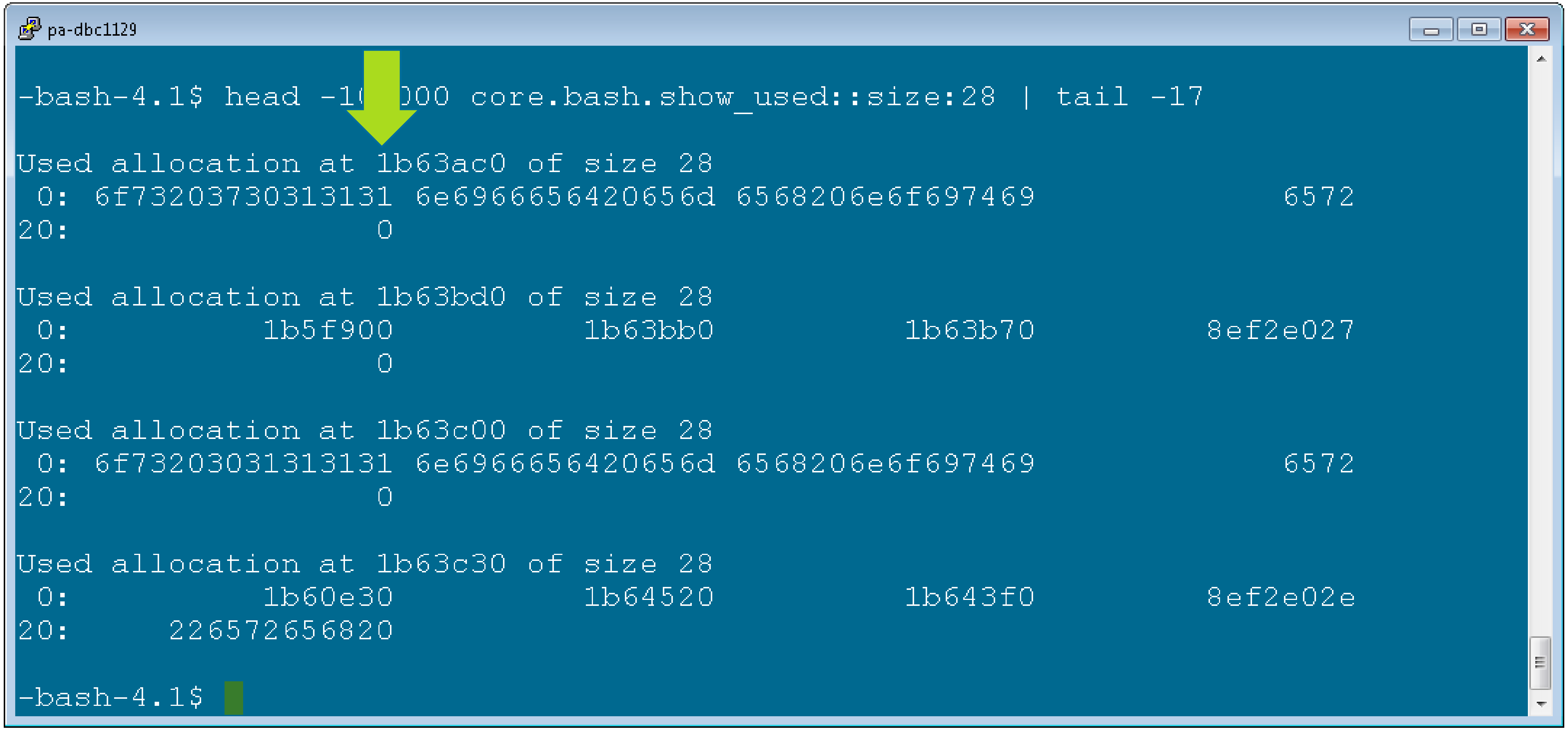
```
-bash-4.1$ chap core.bash
> count used
1364661 allocations use 0x3049d88 (50,634,120) bytes.
> count free
5 allocations use 0x11ca0 (72,864) bytes.
> count stacks
1 stacks use 0x15000 (86,016) bytes.
> count leaked
0 allocations use 0x0 (0) bytes.
> redirect on
> summarize used
Wrote results to core.bash.summarize_used
> show used /size 28
Wrote results to core.bash.show_used::size:28
>
```

Showing Many Allocations to a File



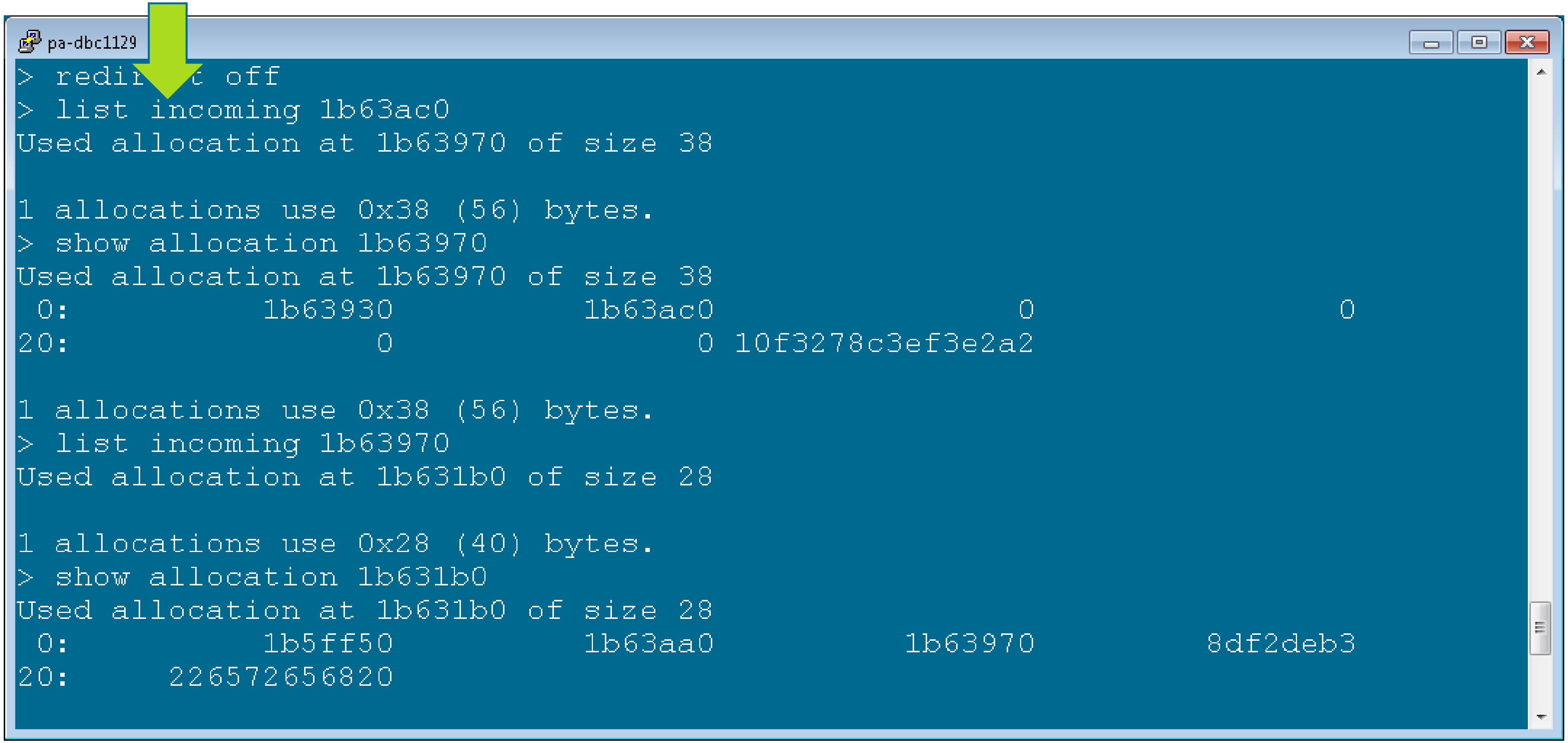
```
-bash-4.1$ chap core.bash
> count used
1364661 allocations use 0x3049d88 (50,634,120) bytes.
> count free
5 allocations use 0x11ca0 (72,864) bytes.
> count stacks
1 stacks use 0x15000 (86,016) bytes.
> count leaked
0 allocations use 0x0 (0) bytes.
> redirect on
> summarize used
Wrote results to core.bash.summarize_used
> show used /size 28
Wrote results to core.bash.show_used::size:28
> 
```

Looking at the Allocations



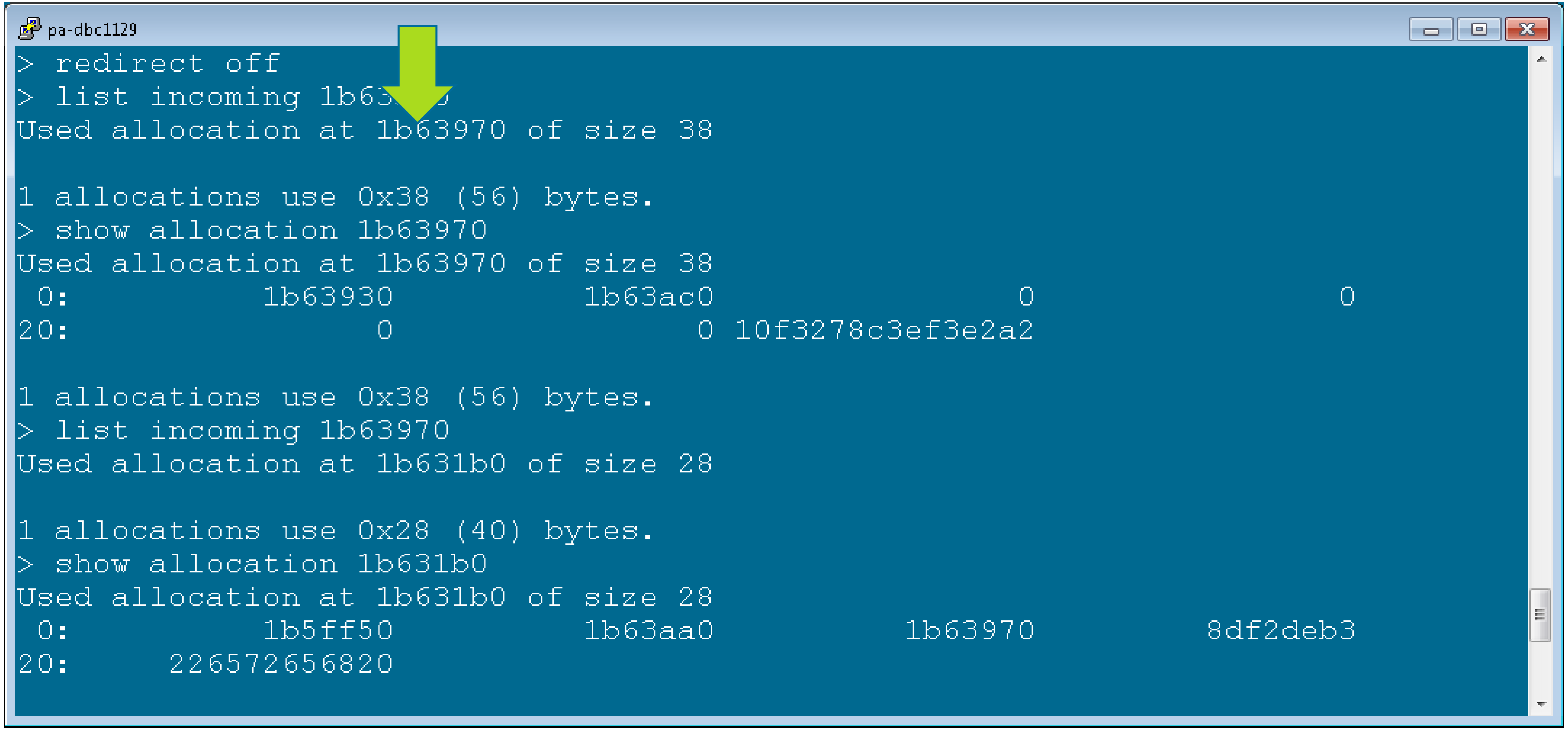
```
-bash-4.1$ head -100000 core.bash.show_used::size:28 | tail -17

Used allocation at 1b63ac0 of size 28
 0: 6f73203730313131 6e6966656420656d 6568206e6f697469             6572
20:                0

Used allocation at 1b63bd0 of size 28
 0:         1b5f900          1b63bb0          1b63b70         8ef2e027
20:                0

Used allocation at 1b63c00 of size 28
 0: 6f73203031313131 6e6966656420656d 6568206e6f697469             6572
20:                0

Used allocation at 1b63c30 of size 28
 0:         1b60e30          1b64520          1b643f0         8ef2e02e
20:      226572656820

-bash-4.1$
```

Following Incoming Edges



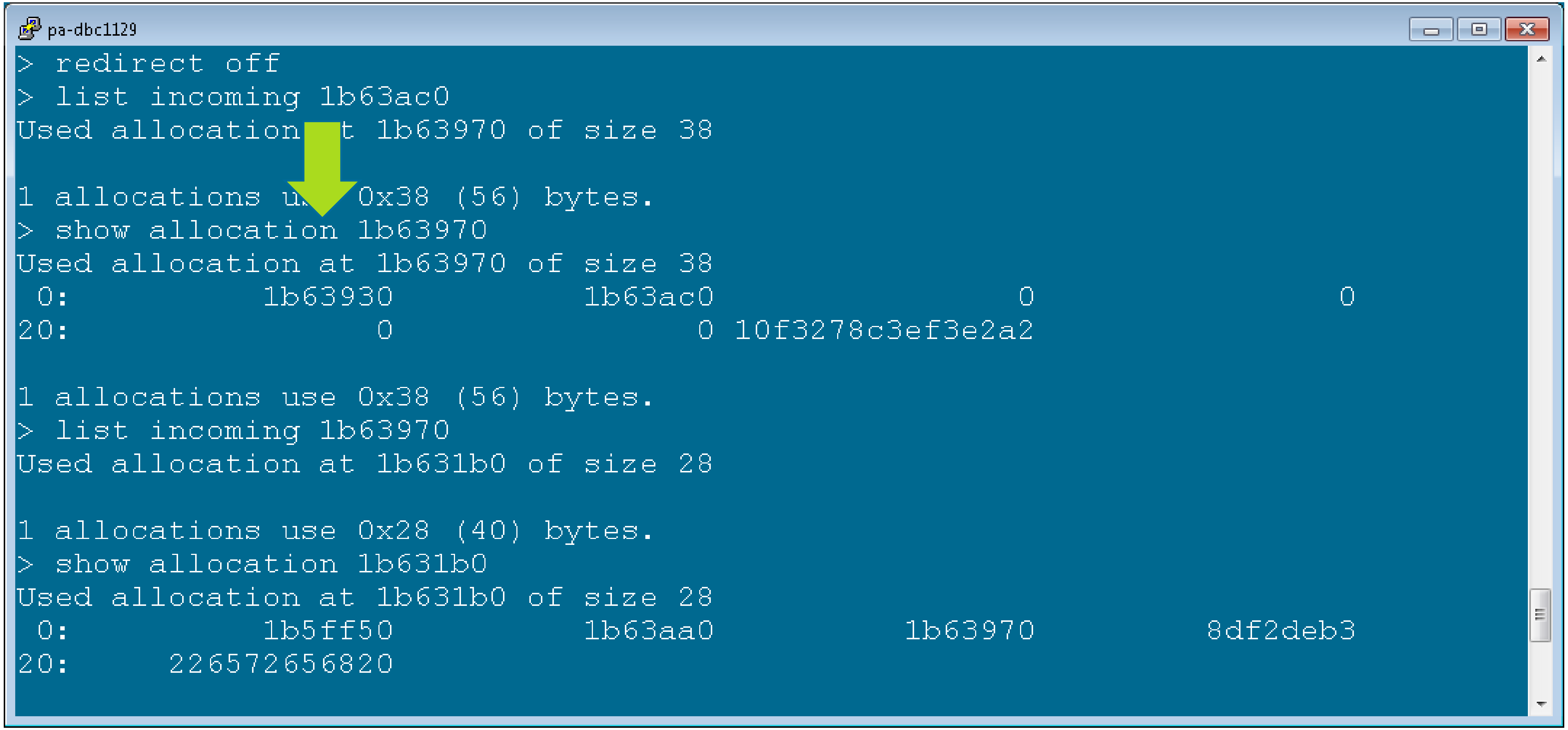
```
> redirect off
> list incoming 1b63ac0
Used allocation at 1b63970 of size 38

1 allocations use 0x38 (56) bytes.
> show allocation 1b63970
Used allocation at 1b63970 of size 38
 0:          1b63930          1b63ac0                0                0
20:                0                0 10f3278c3ef3e2a2

1 allocations use 0x38 (56) bytes.
> list incoming 1b63970
Used allocation at 1b631b0 of size 28

1 allocations use 0x28 (40) bytes.
> show allocation 1b631b0
Used allocation at 1b631b0 of size 28
 0:          1b5ff50          1b63aa0          1b63970         8df2deb3
20:     226572656820
```

Following Incoming Edges



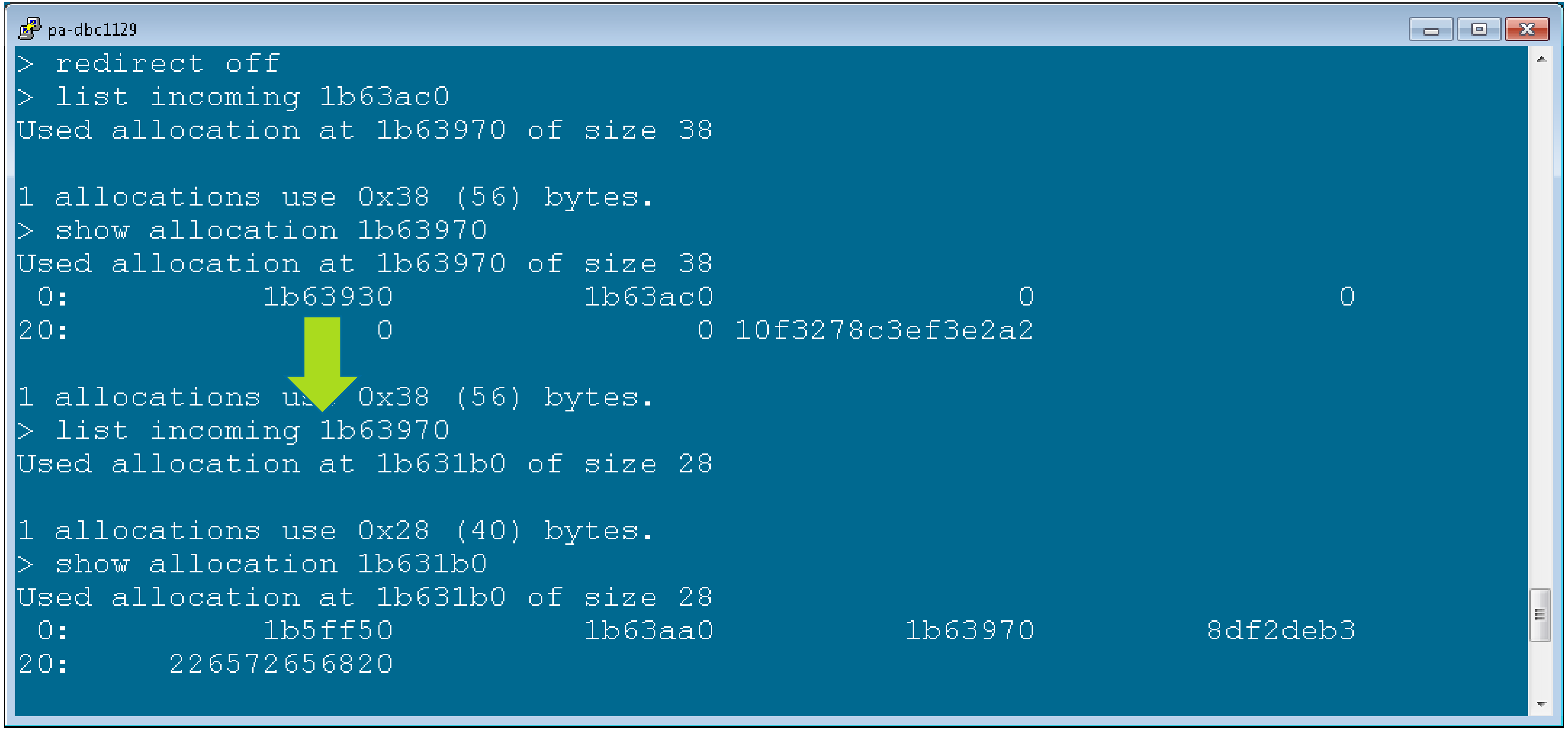
```
> redirect off
> list incoming 1b63970
Used allocation at 1b63970 of size 38

1 allocations use 0x38 (56) bytes.
> show allocation 1b63970
Used allocation at 1b63970 of size 38
 0:          1b63930          1b63ac0                0                0
20:                0                0 10f3278c3ef3e2a2

1 allocations use 0x38 (56) bytes.
> list incoming 1b63970
Used allocation at 1b631b0 of size 28

1 allocations use 0x28 (40) bytes.
> show allocation 1b631b0
Used allocation at 1b631b0 of size 28
 0:          1b5ff50          1b63aa0          1b63970         8df2deb3
20:      22657265682 0
```

Following Incoming Edges



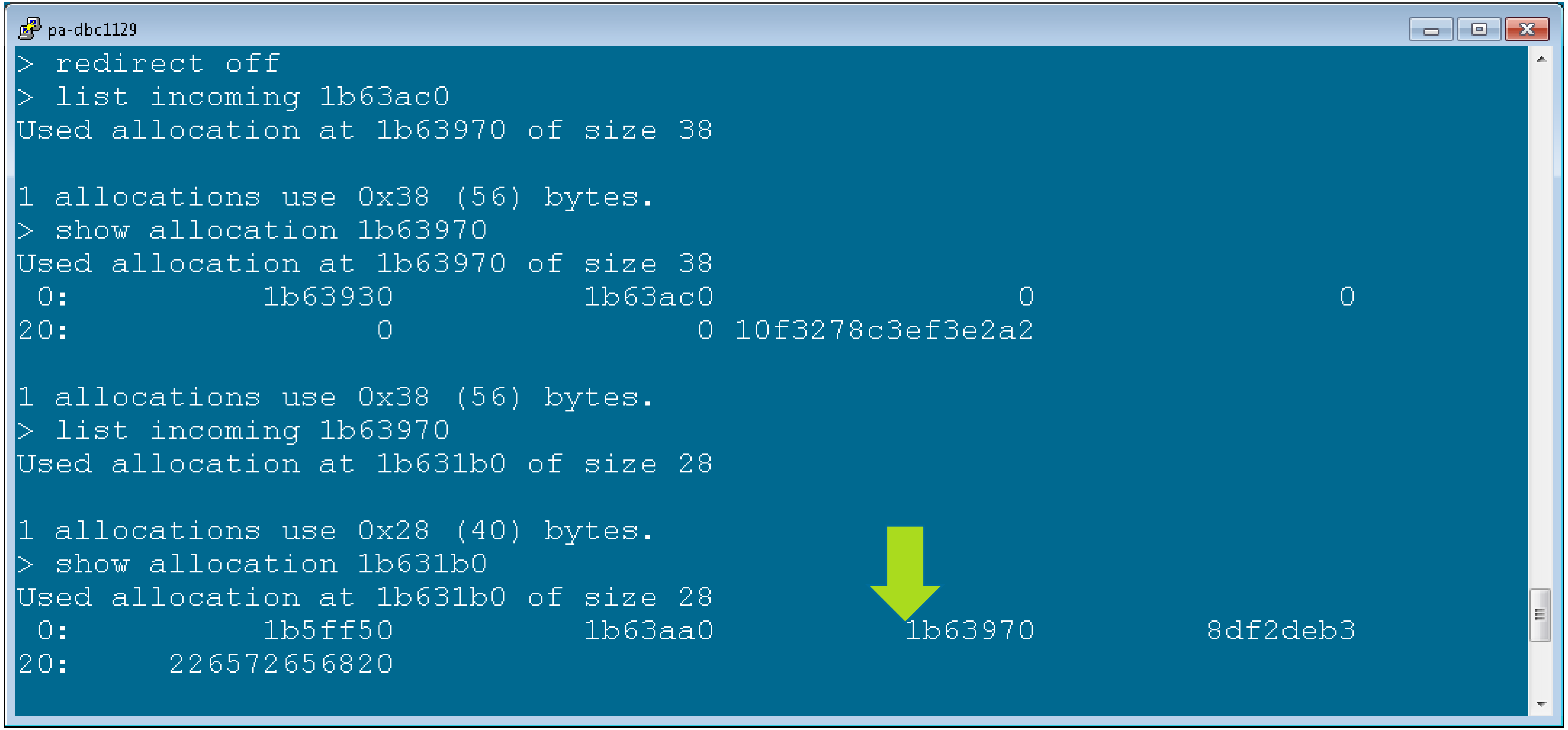
```
> redirect off
> list incoming 1b63ac0
Used allocation at 1b63970 of size 38

1 allocations use 0x38 (56) bytes.
> show allocation 1b63970
Used allocation at 1b63970 of size 38
 0:          1b63930          1b63ac0                0                0
20:                0                0 10f3278c3ef3e2a2

1 allocations use 0x38 (56) bytes.
> list incoming 1b63970
Used allocation at 1b631b0 of size 28

1 allocations use 0x28 (40) bytes.
> show allocation 1b631b0
Used allocation at 1b631b0 of size 28
 0:          1b5ff50          1b63aa0          1b63970         8df2deb3
20:     226572656820
```

Following Incoming Edges



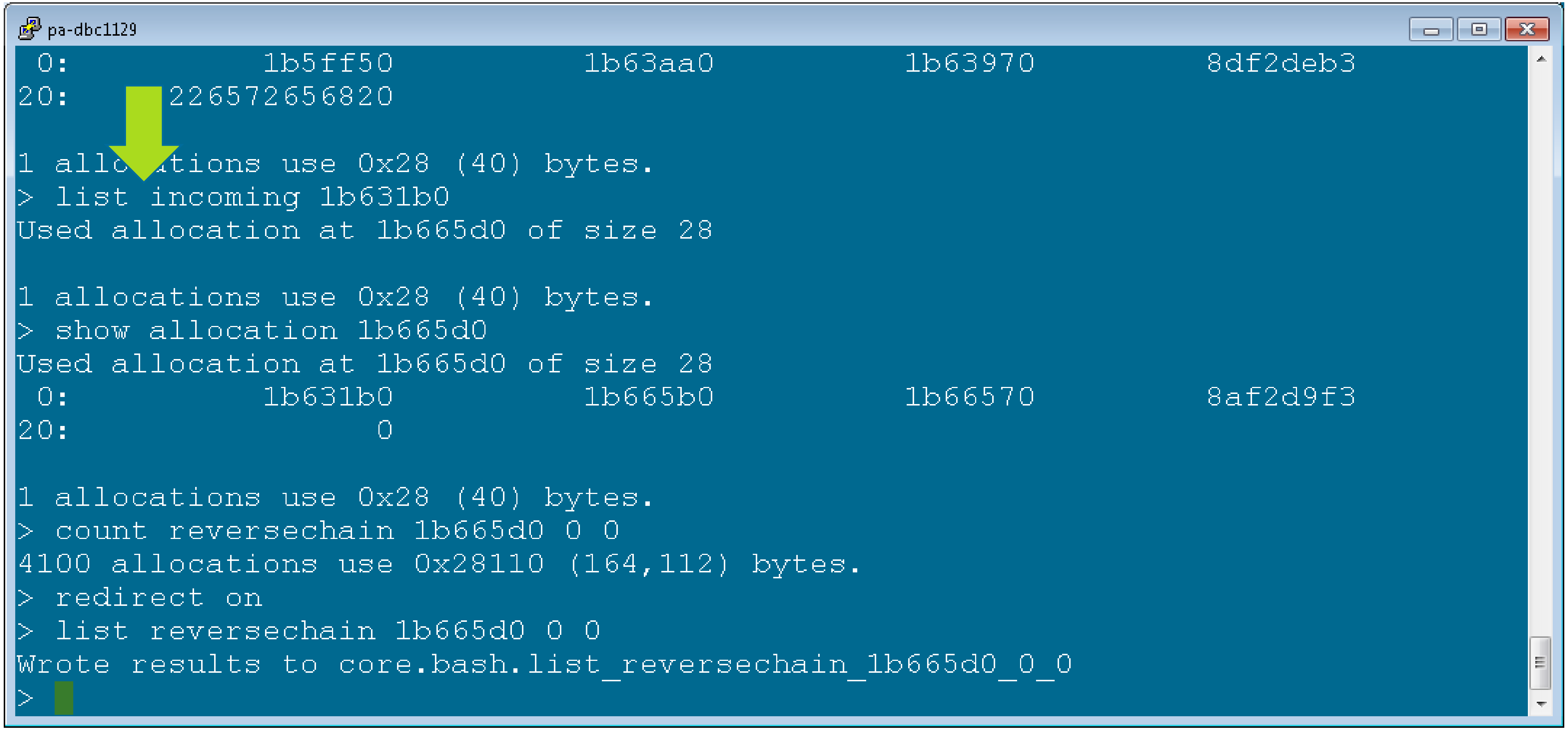
```
> redirect off
> list incoming 1b63ac0
Used allocation at 1b63970 of size 38

1 allocations use 0x38 (56) bytes.
> show allocation 1b63970
Used allocation at 1b63970 of size 38
 0:          1b63930          1b63ac0                0                0
20:                0                0 10f3278c3ef3e2a2

1 allocations use 0x38 (56) bytes.
> list incoming 1b63970
Used allocation at 1b631b0 of size 28

1 allocations use 0x28 (40) bytes.
> show allocation 1b631b0
Used allocation at 1b631b0 of size 28
 0:          1b5ff50          1b63aa0          1b63970         8df2deb3
20:     226572656820
```

Following Incoming Edges



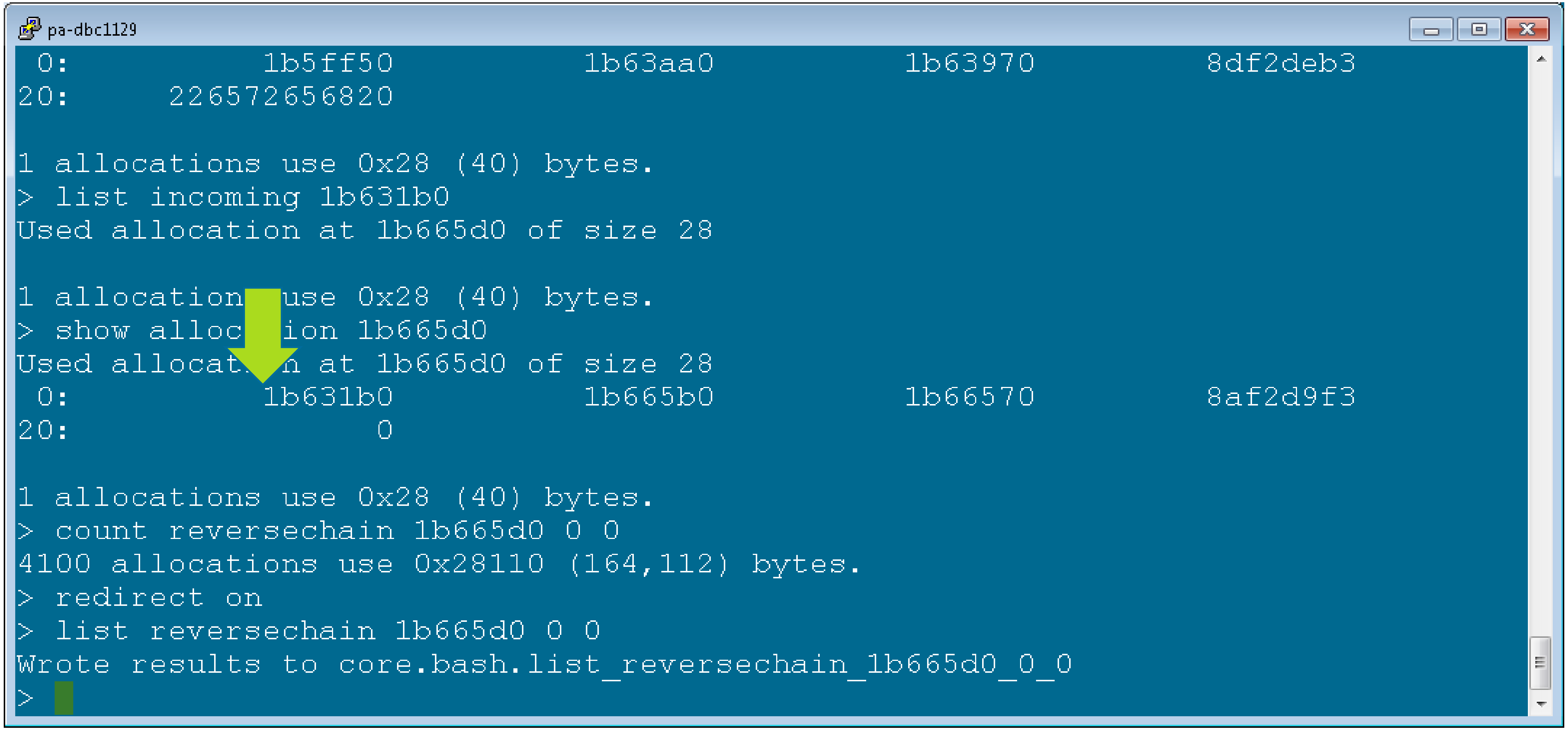
```
> redirect off
> list incoming 1b63ac0
Used allocation at 1b63970 of size 38

1 allocations use 0x38 (56) bytes.
> show allocation 1b63970
Used allocation at 1b63970 of size 38
 0:          1b63930          1b63ac0                0                0
20:                0                0 10f3278c3ef3e2a2

1 allocations use 0x38 (56) bytes.
> list incoming 1b63970
Used allocation at 1b631b0 of size 28

1 allocations use 0x28 (40) bytes.
> show allocation 1b631b0
Used allocation at 1b631b0 of size 28
 0:          1b5ff50          1b63aa0          1b63970         8df2deb3
20:     226572656820
```

Speeding the Traversal



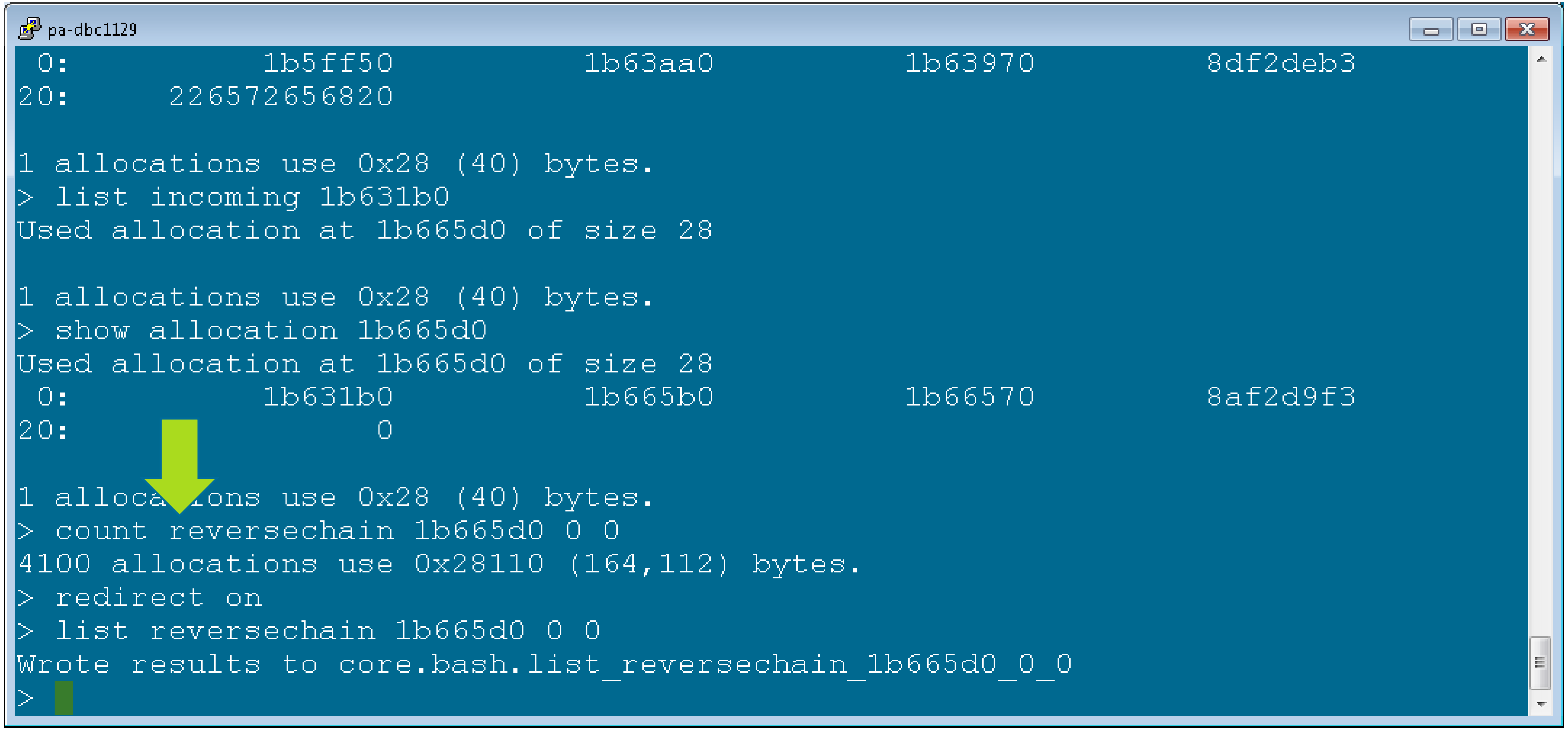
```
 0:          1b5ff50          1b63aa0          1b63970          8df2deb3
20:     226572656820

1 allocations use 0x28 (40) bytes.
> list incoming 1b631b0
Used allocation at 1b665d0 of size 28

1 allocations use 0x28 (40) bytes.
> show allocation 1b665d0
Used allocation at 1b665d0 of size 28
 0:          1b631b0          1b665b0          1b66570          8af2d9f3
20:                0

1 allocations use 0x28 (40) bytes.
> count reversechain 1b665d0 0 0
4100 allocations use 0x28110 (164,112) bytes.
> redirect on
> list reversechain 1b665d0 0 0
Wrote results to core.bash.list_reversechain_1b665d0_0_0
> 
```

Speeding the Traversal



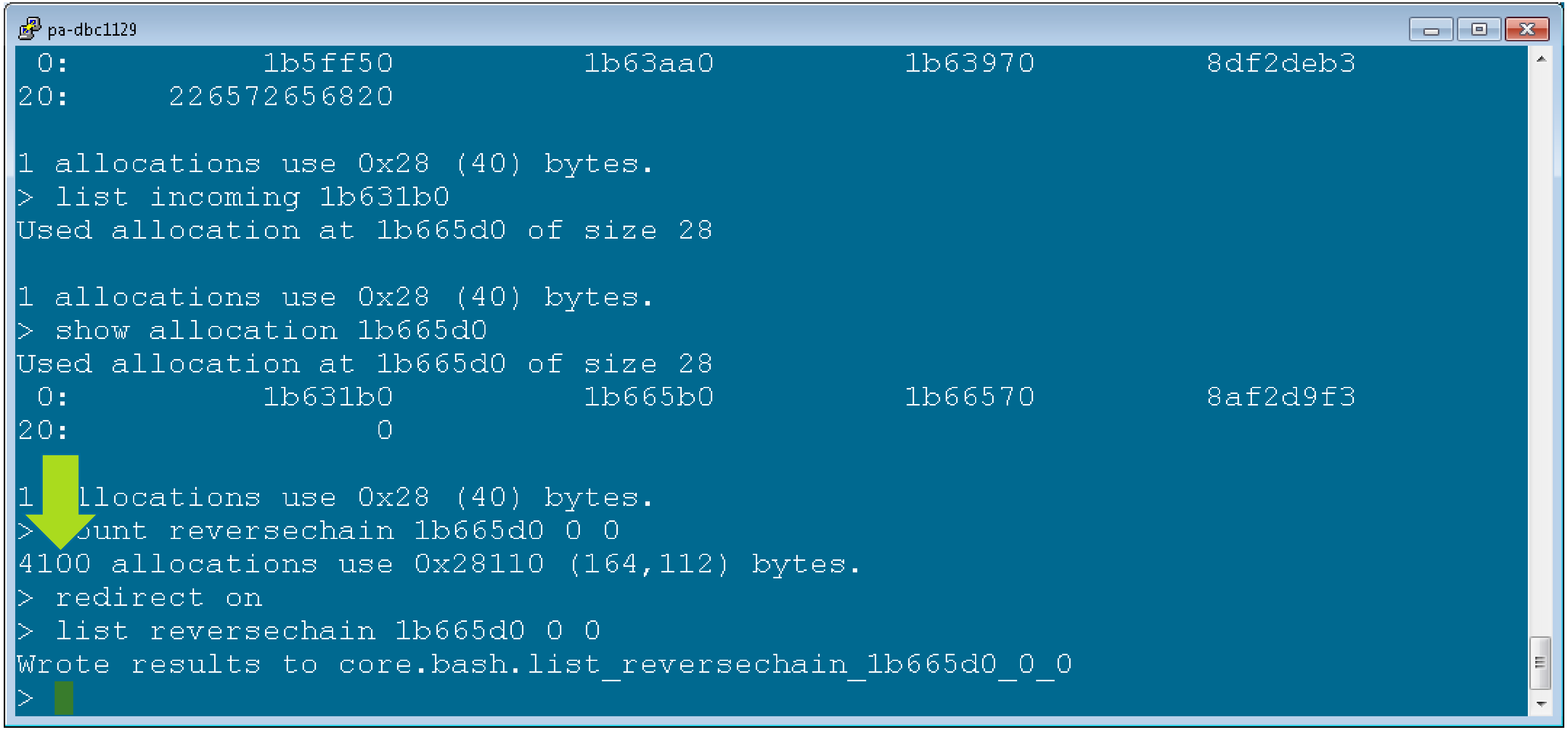
```
 0:          1b5ff50          1b63aa0          1b63970          8df2deb3
20:      226572656820

1 allocations use 0x28 (40) bytes.
> list incoming 1b631b0
Used allocation at 1b665d0 of size 28

1 allocations use 0x28 (40) bytes.
> show allocation 1b665d0
Used allocation at 1b665d0 of size 28
 0:          1b631b0          1b665b0          1b66570          8af2d9f3
20:                0

1 allocations use 0x28 (40) bytes.
> count reversechain 1b665d0 0 0
4100 allocations use 0x28110 (164,112) bytes.
> redirect on
> list reversechain 1b665d0 0 0
Wrote results to core.bash.list_reversechain_1b665d0_0_0
> 
```

Speeding the Traversal



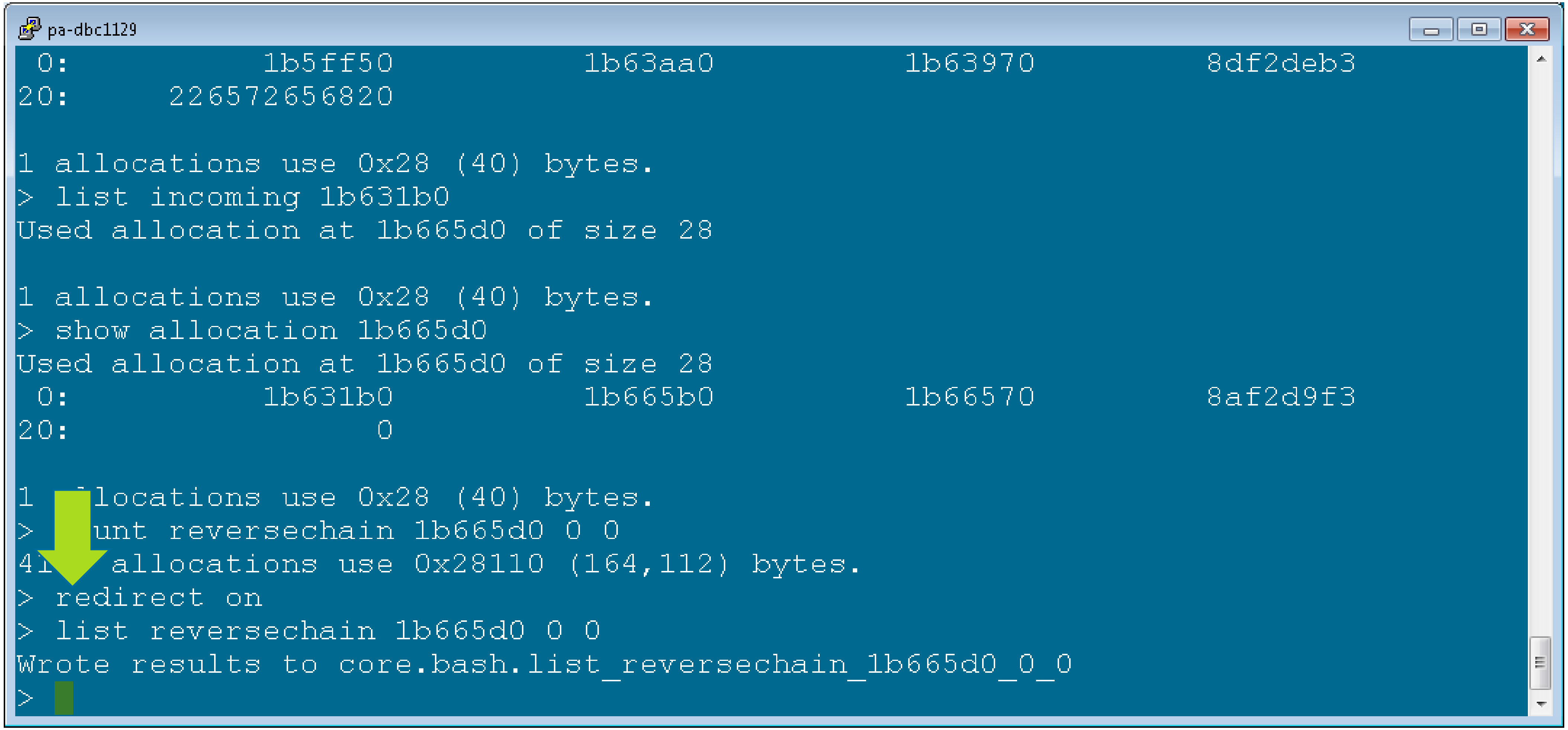
```
 0:            1b5ff50          1b63aa0          1b63970         8df2deb3
20:     226572656820

1 allocations use 0x28 (40) bytes.
> list incoming 1b631b0
Used allocation at 1b665d0 of size 28

1 allocations use 0x28 (40) bytes.
> show allocation 1b665d0
Used allocation at 1b665d0 of size 28
 0:            1b631b0          1b665b0          1b66570         8af2d9f3
20:                0

1 allocations use 0x28 (40) bytes.
> count reversechain 1b665d0 0 0
4100 allocations use 0x28110 (164,112) bytes.
> redirect on
> list reversechain 1b665d0 0 0
Wrote results to core.bash.list_reversechain_1b665d0_0_0
> 
```

Speeding the Traversal



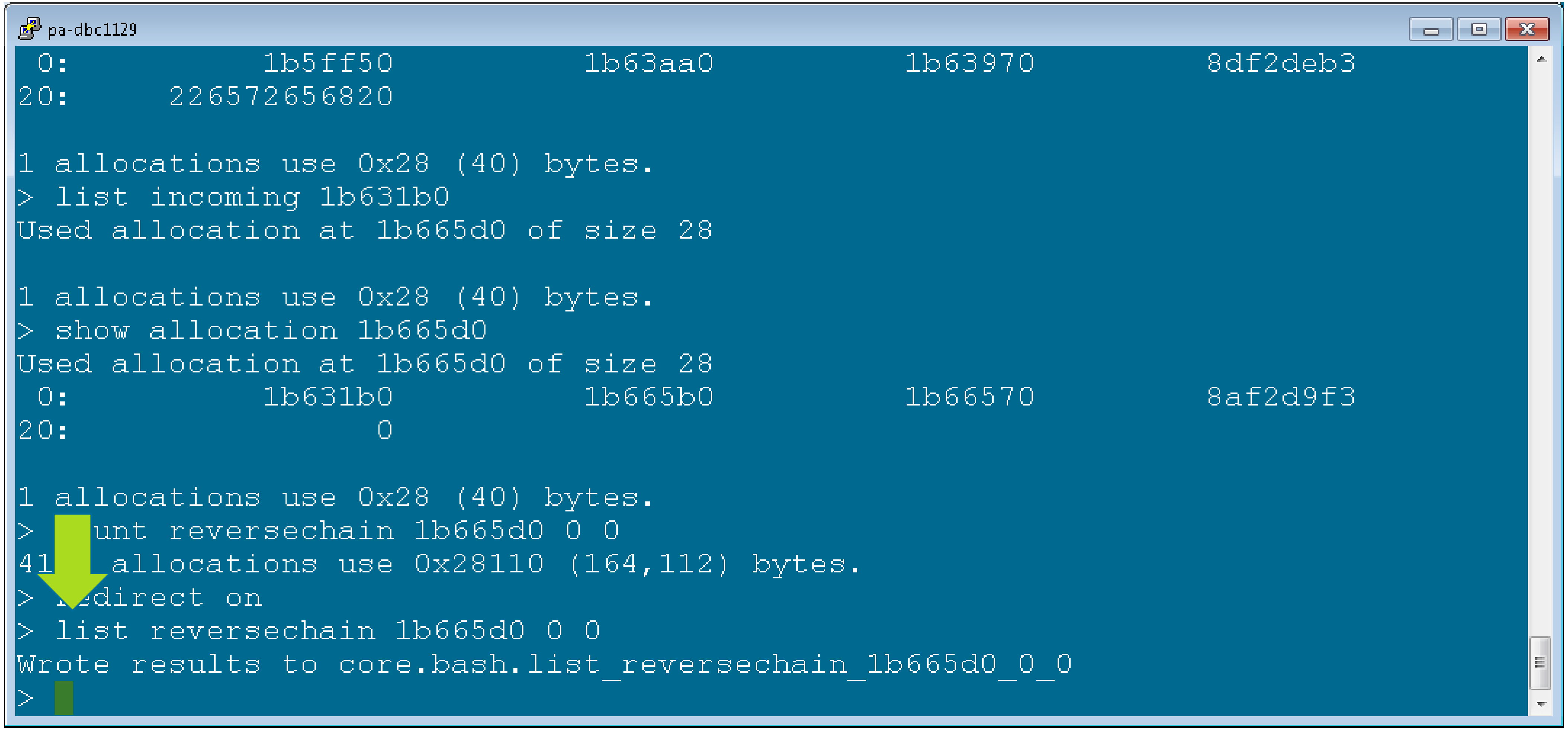
```
 0:          1b5ff50          1b63aa0          1b63970         8df2deb3
20:     226572656820

1 allocations use 0x28 (40) bytes.
> list incoming 1b631b0
Used allocation at 1b665d0 of size 28

1 allocations use 0x28 (40) bytes.
> show allocation 1b665d0
Used allocation at 1b665d0 of size 28
 0:          1b631b0          1b665b0          1b66570         8af2d9f3
20:                0

1  llocations use 0x28 (40) bytes.
>  ount reversechain 1b665d0 0 0
4100 allocations use 0x28110 (164,112) bytes.
> redirect on
> list reversechain 1b665d0 0 0
Wrote results to core.bash.list_reversechain_1b665d0_0_0
>
```

Speeding the Traversal



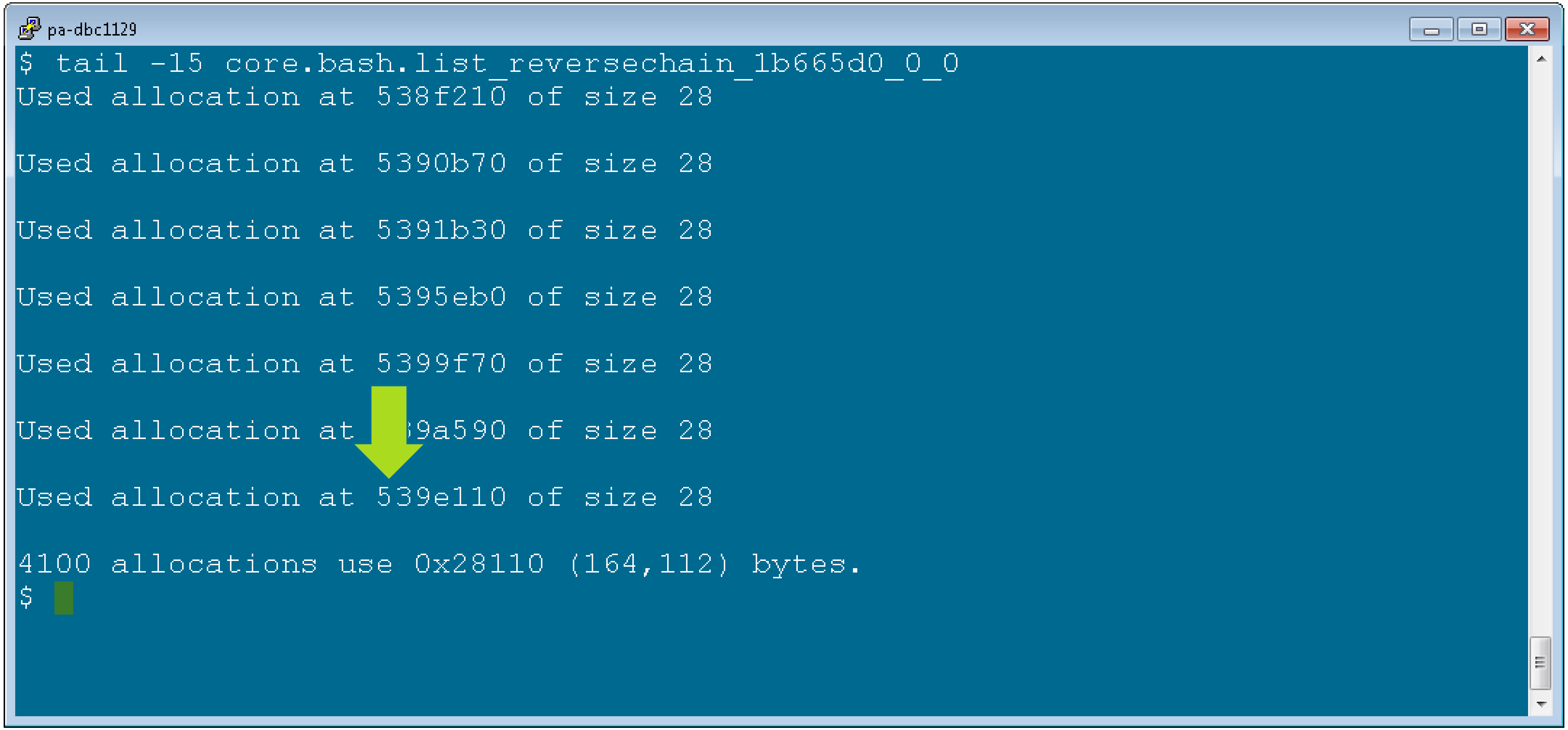
```
 0:          1b5ff50          1b63aa0          1b63970          8df2deb3
20:     226572656820

1 allocations use 0x28 (40) bytes.
> list incoming 1b631b0
Used allocation at 1b665d0 of size 28

1 allocations use 0x28 (40) bytes.
> show allocation 1b665d0
Used allocation at 1b665d0 of size 28
 0:          1b631b0          1b665b0          1b66570          8af2d9f3
20:                0

1   llocations use 0x28 (40) bytes.
>   unt reversechain 1b665d0 0 0
41    allocations use 0x28110 (164,112) bytes.
> redirect on
> list reversechain 1b665d0 0 0
Wrote results to core.bash.list_reversechain_1b665d0_0_0
> 
```

Speeding the Traversal



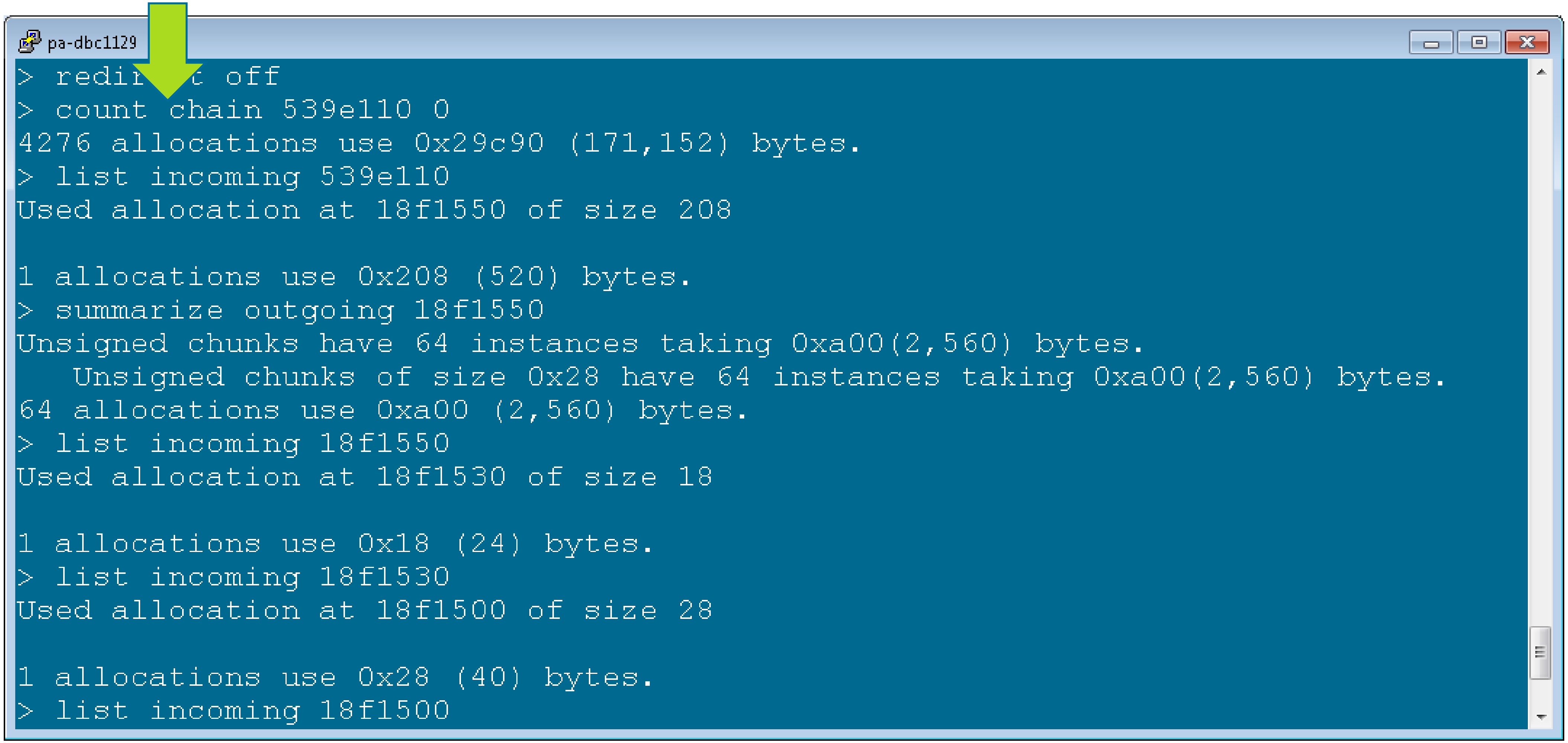
```
 0:          1b5ff50          1b63aa0          1b63970         8df2deb3
20:      226572656820

1 allocations use 0x28 (40) bytes.
> list incoming 1b631b0
Used allocation at 1b665d0 of size 28

1 allocations use 0x28 (40) bytes.
> show allocation 1b665d0
Used allocation at 1b665d0 of size 28
 0:          1b631b0          1b665b0          1b66570         8af2d9f3
20:                0

1 allocations use 0x28 (40) bytes.
> count reversechain 1b665d0 0 0
41[illegible] allocations use 0x28110 (164,112) bytes.
> redirect on
> list reversechain 1b665d0 0 0
Wrote results to core.bash.list_reversechain_1b665d0_0_0
> 
```

The Start of the Chain



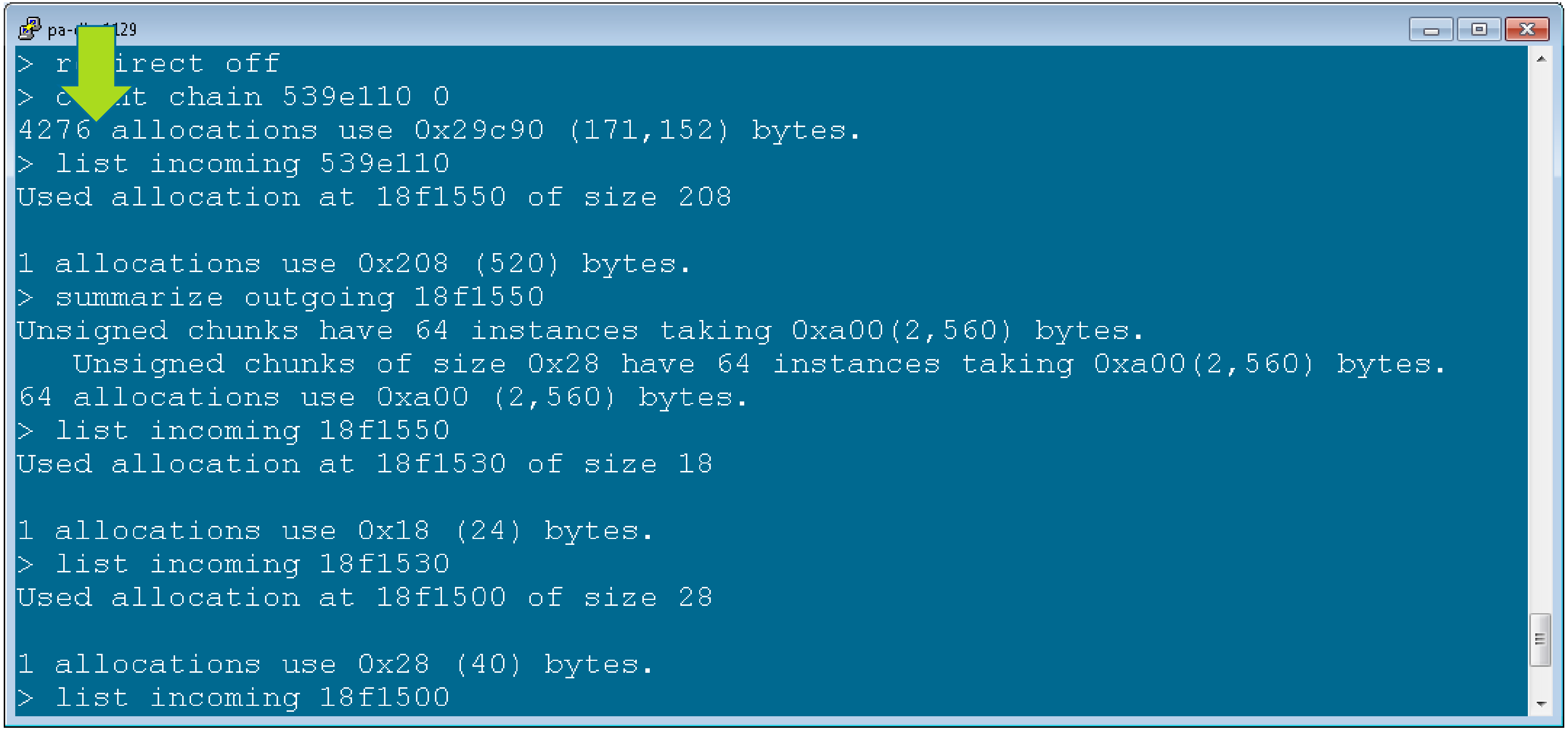
```
$ tail -15 core.bash.list_reversechain_1b665d0_0_0
Used allocation at 538f210 of size 28

Used allocation at 5390b70 of size 28

Used allocation at 5391b30 of size 28

Used allocation at 5395eb0 of size 28

Used allocation at 5399f70 of size 28

Used allocation at 539a590 of size 28

Used allocation at 539e110 of size 28

4100 allocations use 0x28110 (164,112) bytes.
$ 
```

Before the Start of the Chain



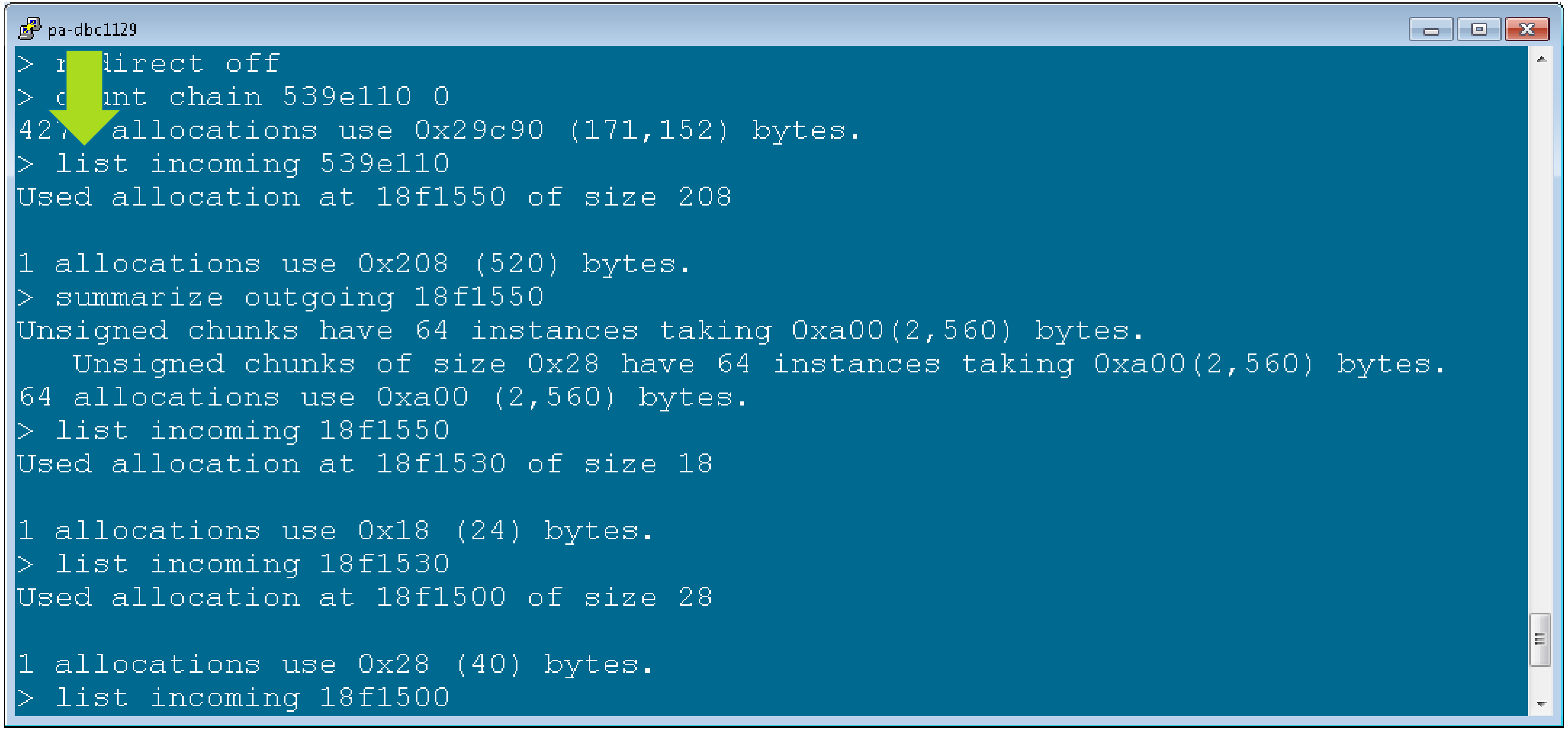
```
> redirect off
> count chain 539e110 0
4276 allocations use 0x29c90 (171,152) bytes.
> list incoming 539e110
Used allocation at 18f1550 of size 208

1 allocations use 0x208 (520) bytes.
> summarize outgoing 18f1550
Unsigned chunks have 64 instances taking 0xa00(2,560) bytes.
   Unsigned chunks of size 0x28 have 64 instances taking 0xa00(2,560) bytes.
64 allocations use 0xa00 (2,560) bytes.
> list incoming 18f1550
Used allocation at 18f1530 of size 18

1 allocations use 0x18 (24) bytes.
> list incoming 18f1530
Used allocation at 18f1500 of size 28

1 allocations use 0x28 (40) bytes.
> list incoming 18f1500
```

Before the Start of the Chain



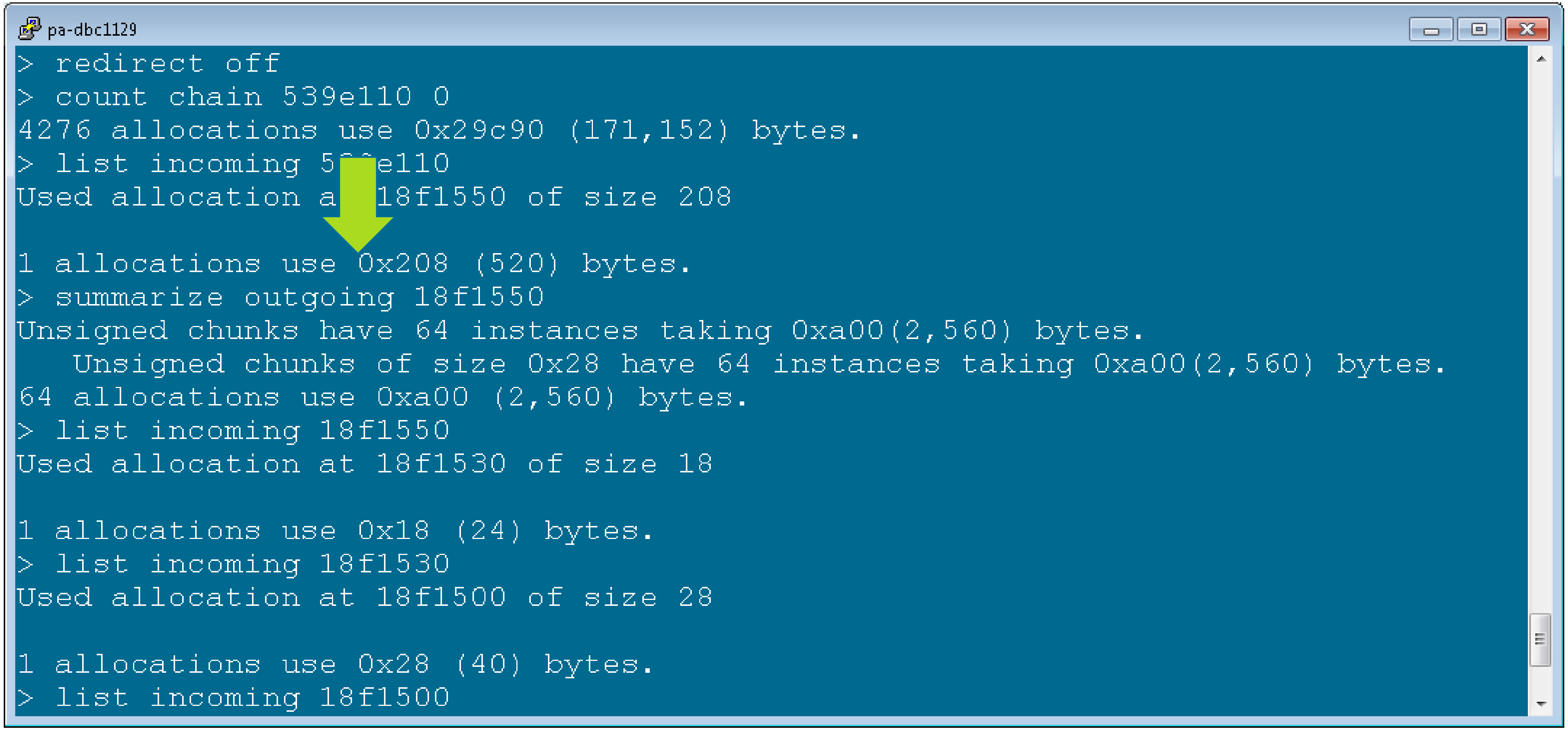
```
> redirect off
> count chain 539e110 0
4276 allocations use 0x29c90 (171,152) bytes.
> list incoming 539e110
Used allocation at 18f1550 of size 208

1 allocations use 0x208 (520) bytes.
> summarize outgoing 18f1550
Unsigned chunks have 64 instances taking 0xa00(2,560) bytes.
   Unsigned chunks of size 0x28 have 64 instances taking 0xa00(2,560) bytes.
64 allocations use 0xa00 (2,560) bytes.
> list incoming 18f1550
Used allocation at 18f1530 of size 18

1 allocations use 0x18 (24) bytes.
> list incoming 18f1530
Used allocation at 18f1500 of size 28

1 allocations use 0x28 (40) bytes.
> list incoming 18f1500
```

Before the Start of the Chain



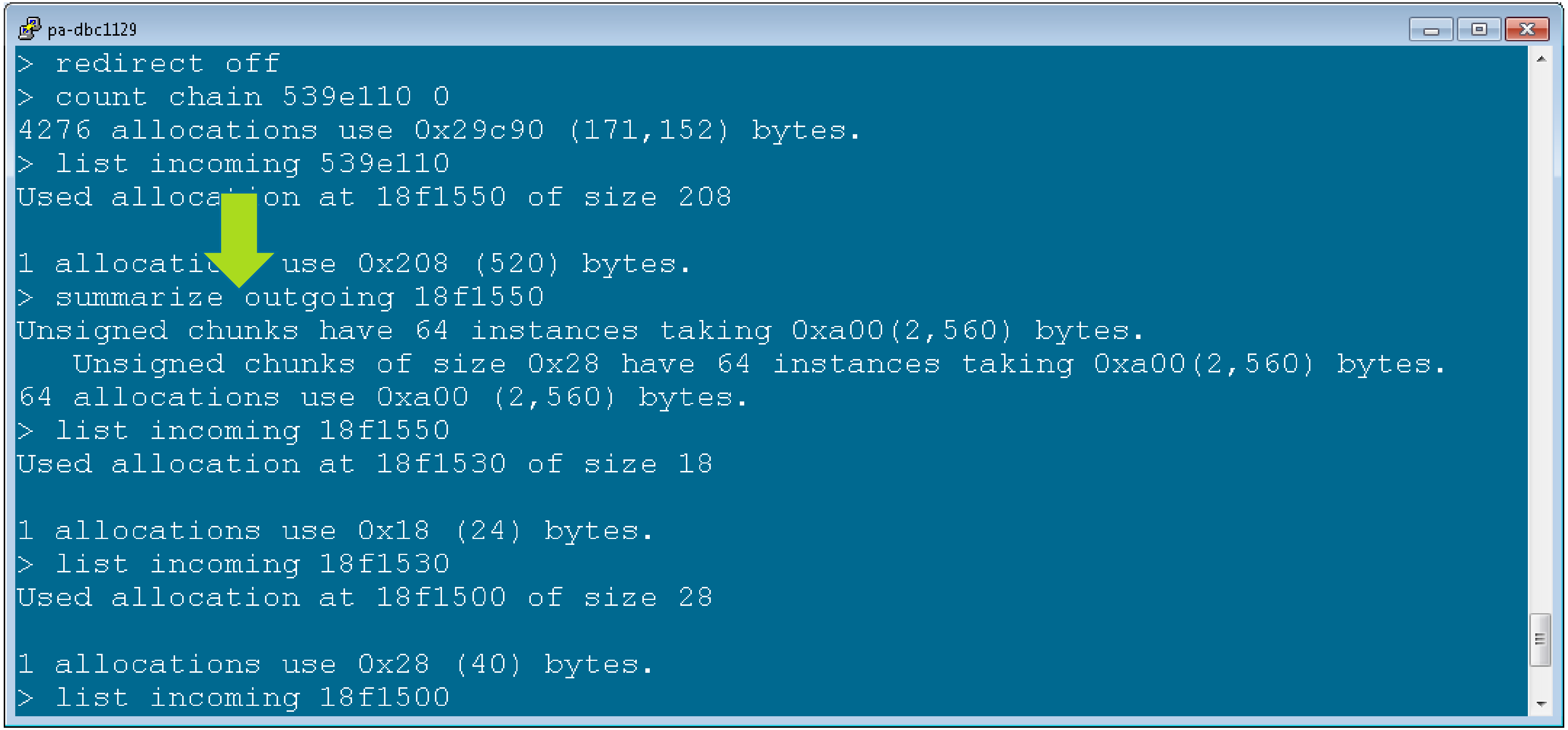
```
> redirect off
> count chain 539e110 0
427 allocations use 0x29c90 (171,152) bytes.
> list incoming 539e110
Used allocation at 18f1550 of size 208

1 allocations use 0x208 (520) bytes.
> summarize outgoing 18f1550
Unsigned chunks have 64 instances taking 0xa00(2,560) bytes.
   Unsigned chunks of size 0x28 have 64 instances taking 0xa00(2,560) bytes.
64 allocations use 0xa00 (2,560) bytes.
> list incoming 18f1550
Used allocation at 18f1530 of size 18

1 allocations use 0x18 (24) bytes.
> list incoming 18f1530
Used allocation at 18f1500 of size 28

1 allocations use 0x28 (40) bytes.
> list incoming 18f1500
```

Before the Start of the Chain



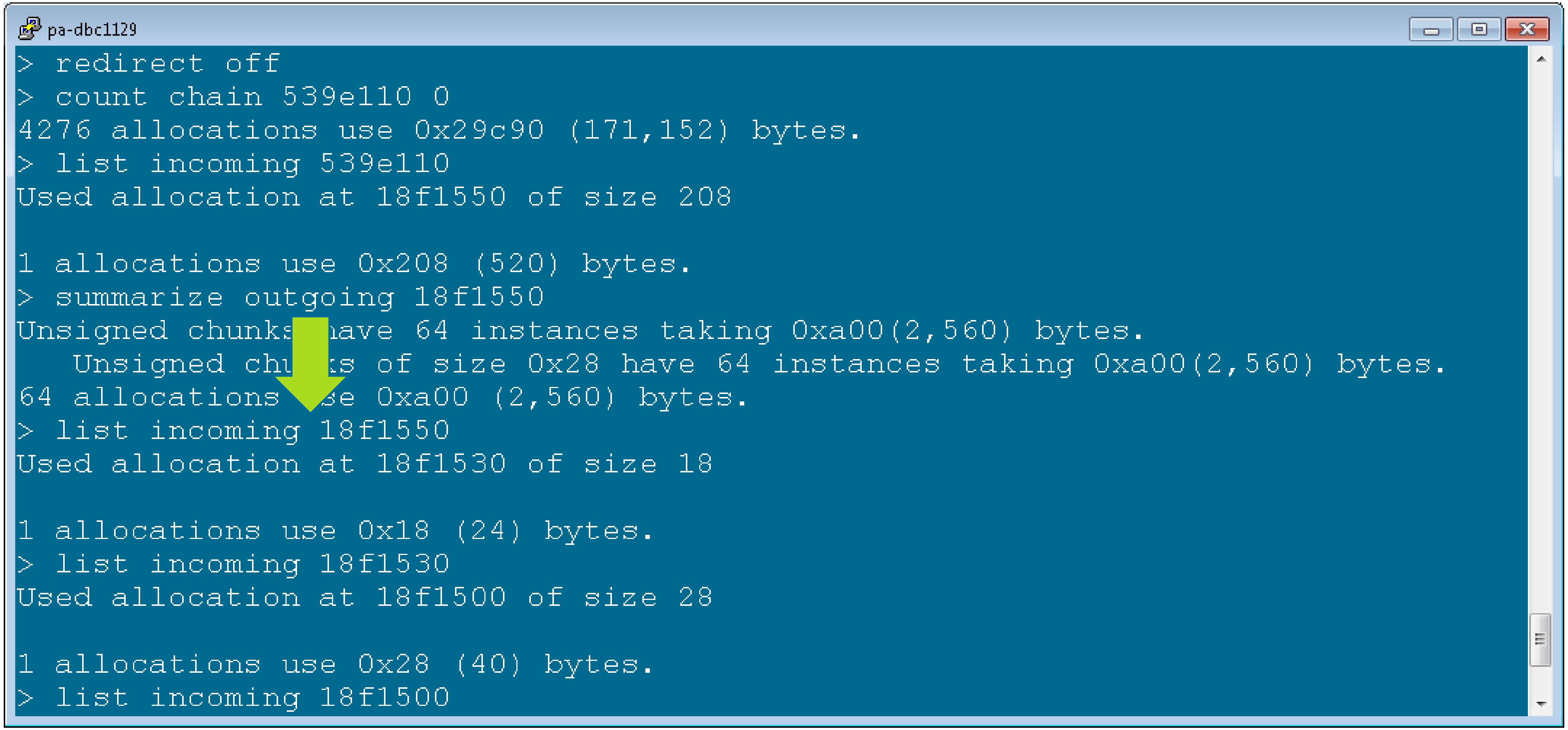
```
> redirect off
> count chain 539e110 0
4276 allocations use 0x29c90 (171,152) bytes.
> list incoming 539e110
Used allocation at 18f1550 of size 208

1 allocations use 0x208 (520) bytes.
> summarize outgoing 18f1550
Unsigned chunks have 64 instances taking 0xa00(2,560) bytes.
   Unsigned chunks of size 0x28 have 64 instances taking 0xa00(2,560) bytes.
64 allocations use 0xa00 (2,560) bytes.
> list incoming 18f1550
Used allocation at 18f1530 of size 18

1 allocations use 0x18 (24) bytes.
> list incoming 18f1530
Used allocation at 18f1500 of size 28

1 allocations use 0x28 (40) bytes.
> list incoming 18f1500
```

Before the Start of the Chain



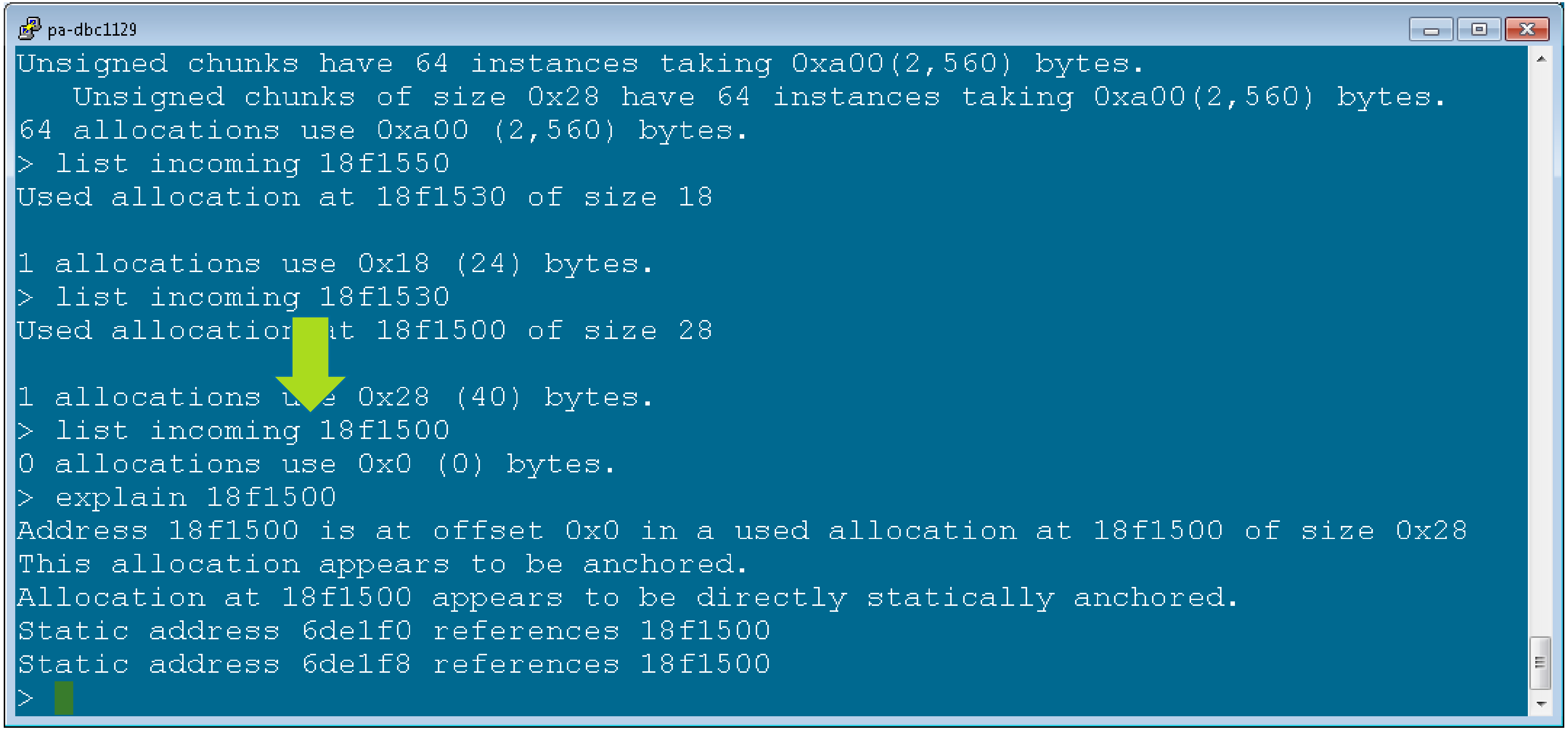
```
> redirect off
> count chain 539e110 0
4276 allocations use 0x29c90 (171,152) bytes.
> list incoming 539e110
Used allocation at 18f1550 of size 208

1 allocations use 0x208 (520) bytes.
> summarize outgoing 18f1550
Unsigned chunks have 64 instances taking 0xa00(2,560) bytes.
   Unsigned chunks of size 0x28 have 64 instances taking 0xa00(2,560) bytes.
64 allocations use 0xa00 (2,560) bytes.
> list incoming 18f1550
Used allocation at 18f1530 of size 18

1 allocations use 0x18 (24) bytes.
> list incoming 18f1530
Used allocation at 18f1500 of size 28

1 allocations use 0x28 (40) bytes.
> list incoming 18f1500
```

Before the Start of the Chain



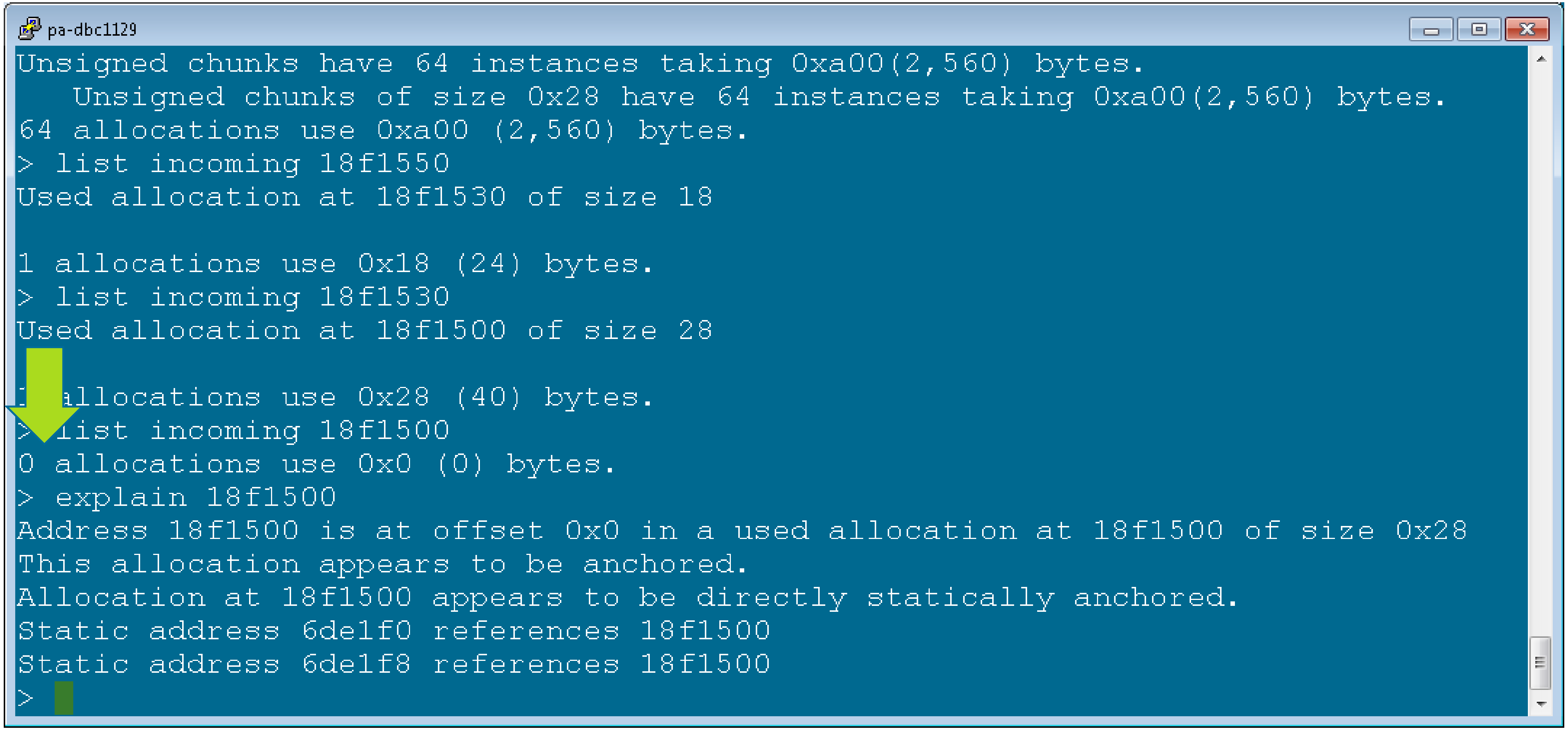
```
> redirect off
> count chain 539e110 0
4276 allocations use 0x29c90 (171,152) bytes.
> list incoming 539e110
Used allocation at 18f1550 of size 208

1 allocations use 0x208 (520) bytes.
> summarize outgoing 18f1550
Unsigned chunks have 64 instances taking 0xa00(2,560) bytes.
   Unsigned chunks of size 0x28 have 64 instances taking 0xa00(2,560) bytes.
64 allocations use 0xa00 (2,560) bytes.
> list incoming 18f1550
Used allocation at 18f1530 of size 18

1 allocations use 0x18 (24) bytes.
> list incoming 18f1530
Used allocation at 18f1500 of size 28

1 allocations use 0x28 (40) bytes.
> list incoming 18f1500
```

Finding the Anchor



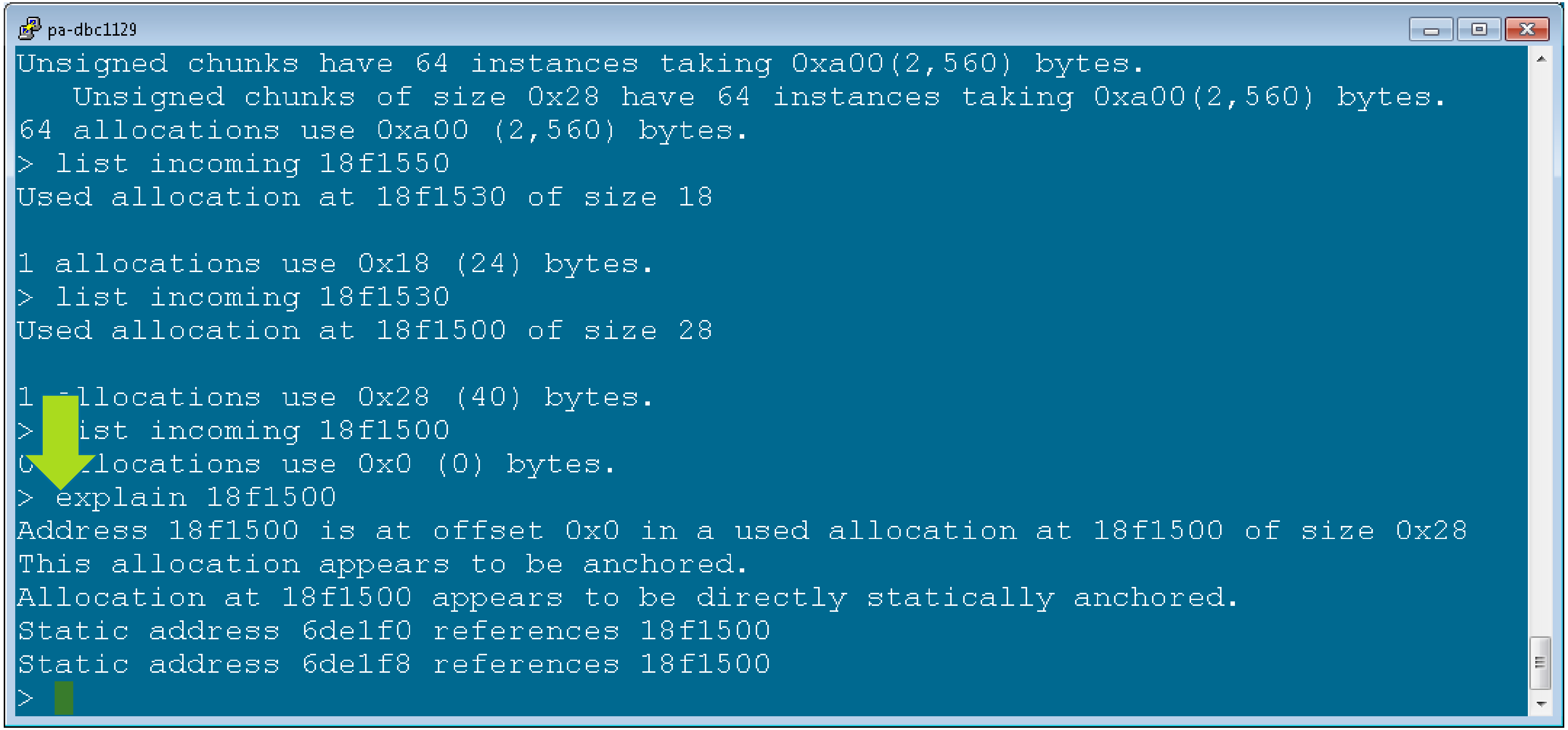
```
Unsigned chunks have 64 instances taking 0xa00(2,560) bytes.
   Unsigned chunks of size 0x28 have 64 instances taking 0xa00(2,560) bytes.
64 allocations use 0xa00 (2,560) bytes.
> list incoming 18f1550
Used allocation at 18f1530 of size 18

1 allocations use 0x18 (24) bytes.
> list incoming 18f1530
Used allocation at 18f1500 of size 28

1 allocations use 0x28 (40) bytes.
> list incoming 18f1500
0 allocations use 0x0 (0) bytes.
> explain 18f1500
Address 18f1500 is at offset 0x0 in a used allocation at 18f1500 of size 0x28
This allocation appears to be anchored.
Allocation at 18f1500 appears to be directly statically anchored.
Static address 6de1f0 references 18f1500
Static address 6de1f8 references 18f1500
>
```

Finding the Anchor



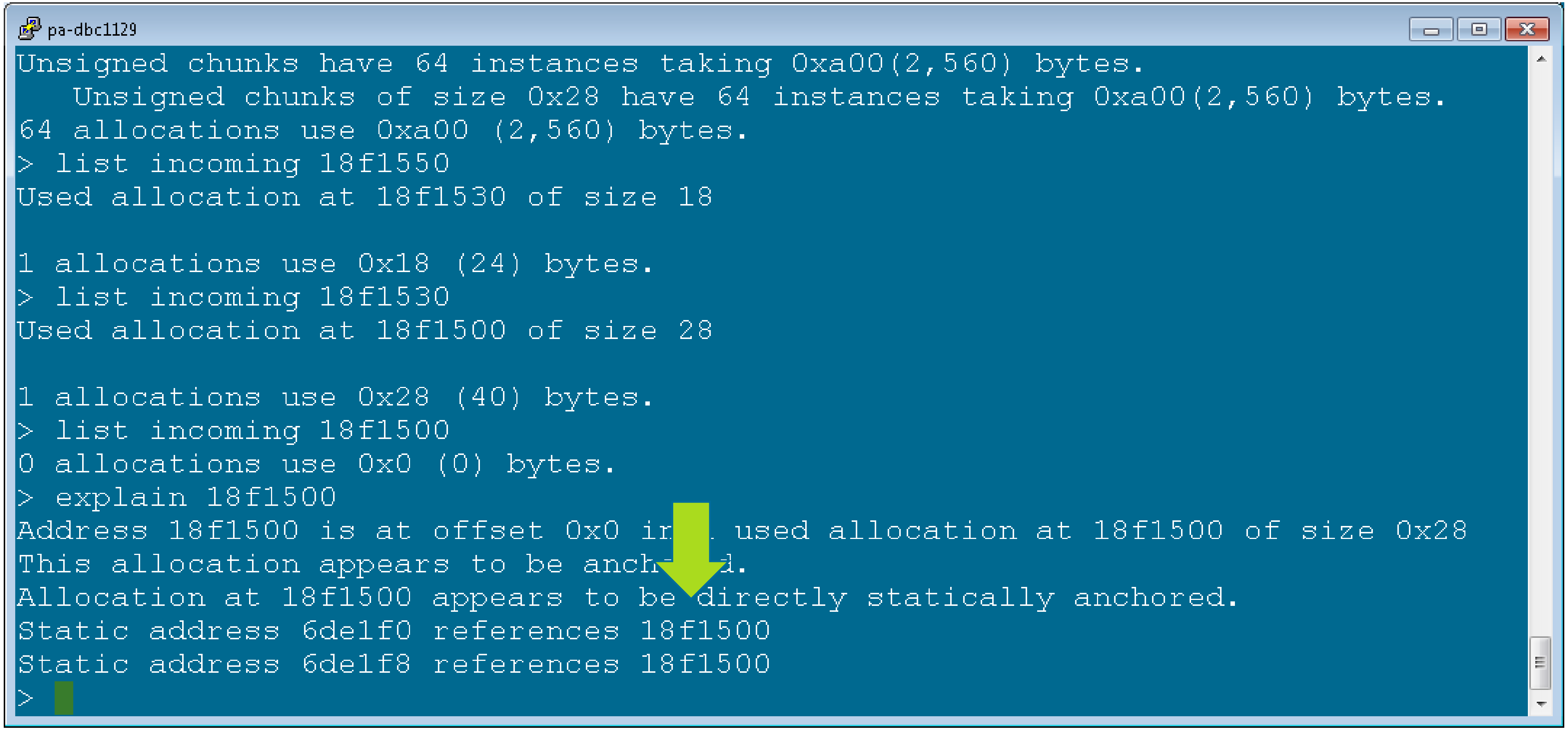
```
Unsigned chunks have 64 instances taking 0xa00(2,560) bytes.
   Unsigned chunks of size 0x28 have 64 instances taking 0xa00(2,560) bytes.
64 allocations use 0xa00 (2,560) bytes.
> list incoming 18f1550
Used allocation at 18f1530 of size 18

1 allocations use 0x18 (24) bytes.
> list incoming 18f1530
Used allocation at 18f1500 of size 28

1 allocations use 0x28 (40) bytes.
> list incoming 18f1500
0 allocations use 0x0 (0) bytes.
> explain 18f1500
Address 18f1500 is at offset 0x0 in a used allocation at 18f1500 of size 0x28
This allocation appears to be anchored.
Allocation at 18f1500 appears to be directly statically anchored.
Static address 6de1f0 references 18f1500
Static address 6de1f8 references 18f1500
>
```

Finding the Anchor



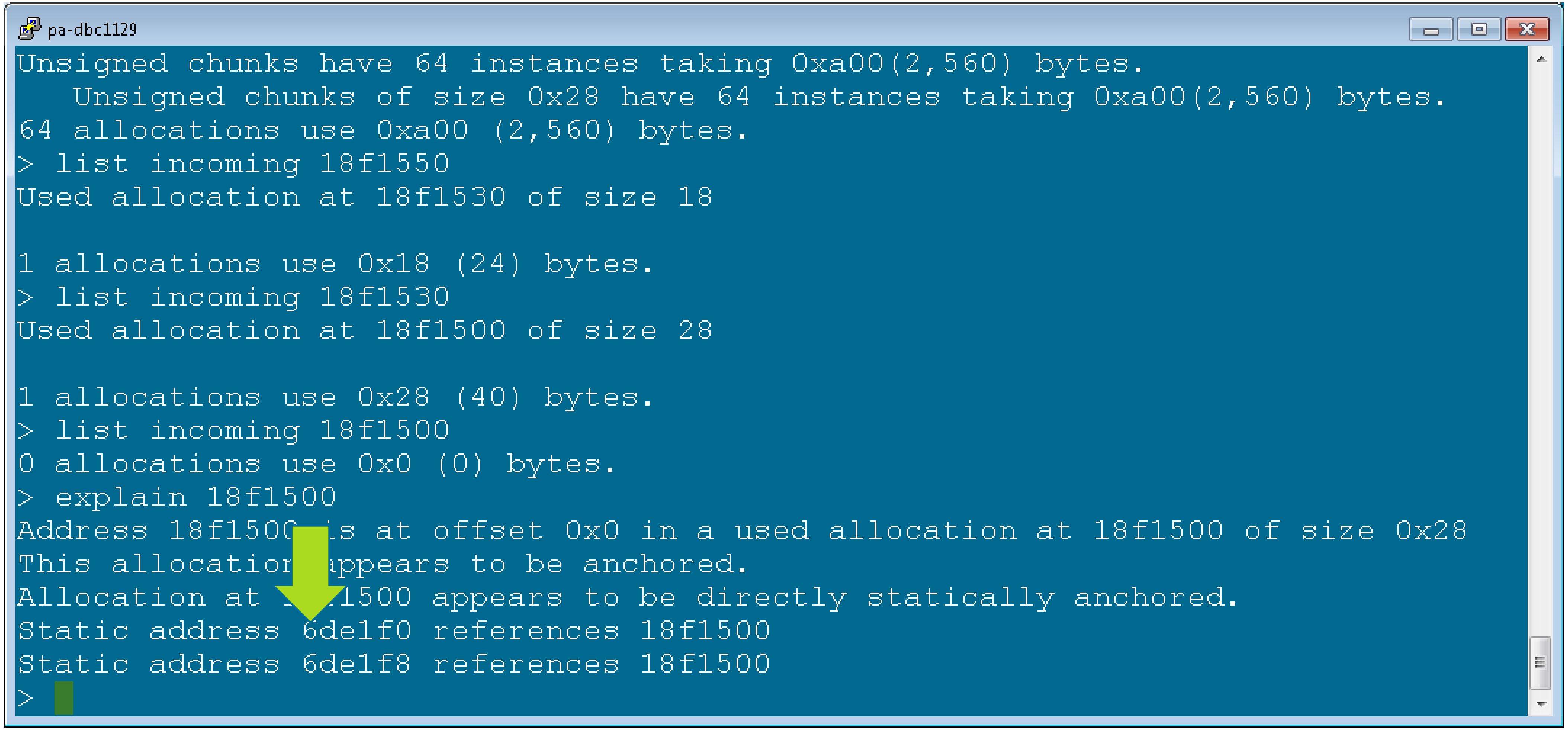
```
Unsigned chunks have 64 instances taking 0xa00(2,560) bytes.
   Unsigned chunks of size 0x28 have 64 instances taking 0xa00(2,560) bytes.
64 allocations use 0xa00 (2,560) bytes.
> list incoming 18f1550
Used allocation at 18f1530 of size 18

1 allocations use 0x18 (24) bytes.
> list incoming 18f1530
Used allocation at 18f1500 of size 28

1 allocations use 0x28 (40) bytes.
> list incoming 18f1500
0 allocations use 0x0 (0) bytes.
> explain 18f1500
Address 18f1500 is at offset 0x0 in a used allocation at 18f1500 of size 0x28
This allocation appears to be anchored.
Allocation at 18f1500 appears to be directly statically anchored.
Static address 6de1f0 references 18f1500
Static address 6de1f8 references 18f1500
>
```

Finding the Anchor



```
Unsigned chunks have 64 instances taking 0xa00(2,560) bytes.
   Unsigned chunks of size 0x28 have 64 instances taking 0xa00(2,560) bytes.
64 allocations use 0xa00 (2,560) bytes.
> list incoming 18f1550
Used allocation at 18f1530 of size 18

1 allocations use 0x18 (24) bytes.
> list incoming 18f1530
Used allocation at 18f1500 of size 28

1 allocations use 0x28 (40) bytes.
> list incoming 18f1500
0 allocations use 0x0 (0) bytes.
> explain 18f1500
Address 18f1500 is at offset 0x0 in a used allocation at 18f1500 of size 0x28
This allocation appears to be anchored.
Allocation at 18f1500 appears to be directly statically anchored.
Static address 6de1f0 references 18f1500
Static address 6de1f8 references 18f1500
>
```

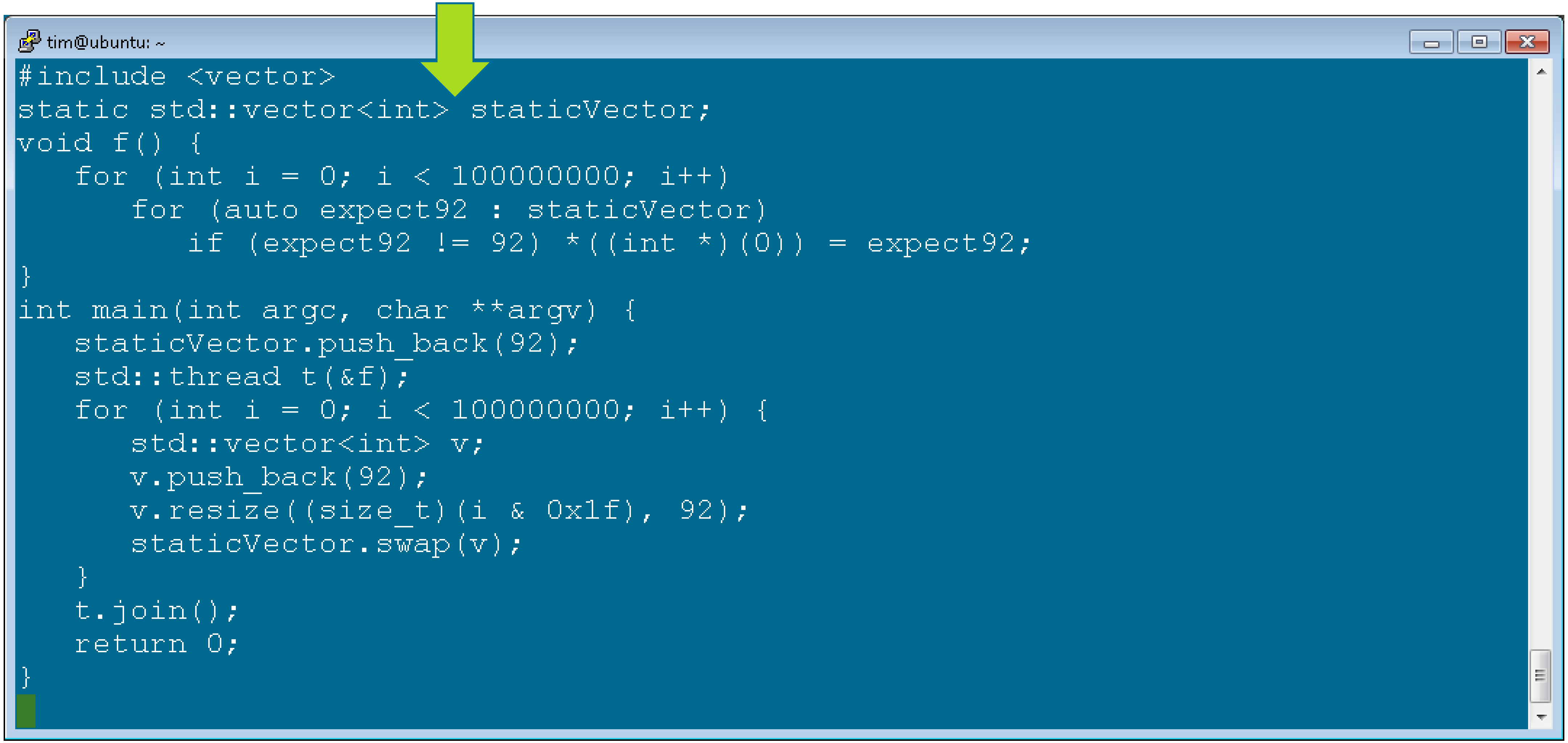
Finding the Anchor



```
Unsigned chunks have 64 instances taking 0xa00(2,560) bytes.
   Unsigned chunks of size 0x28 have 64 instances taking 0xa00(2,560) bytes.
64 allocations use 0xa00 (2,560) bytes.
> list incoming 18f1550
Used allocation at 18f1530 of size 18

1 allocations use 0x18 (24) bytes.
> list incoming 18f1530
Used allocation at 18f1500 of size 28

1 allocations use 0x28 (40) bytes.
> list incoming 18f1500
0 allocations use 0x0 (0) bytes.
> explain 18f1500
Address 18f1500 is at offset 0x0 in a used allocation at 18f1500 of size 0x28
This allocation appears to be anchored.
Allocation at 18f1500 appears to be directly statically anchored.
Static address 6de1f0 references 18f1500
Static address 6de1f8 references 18f1500
>
```

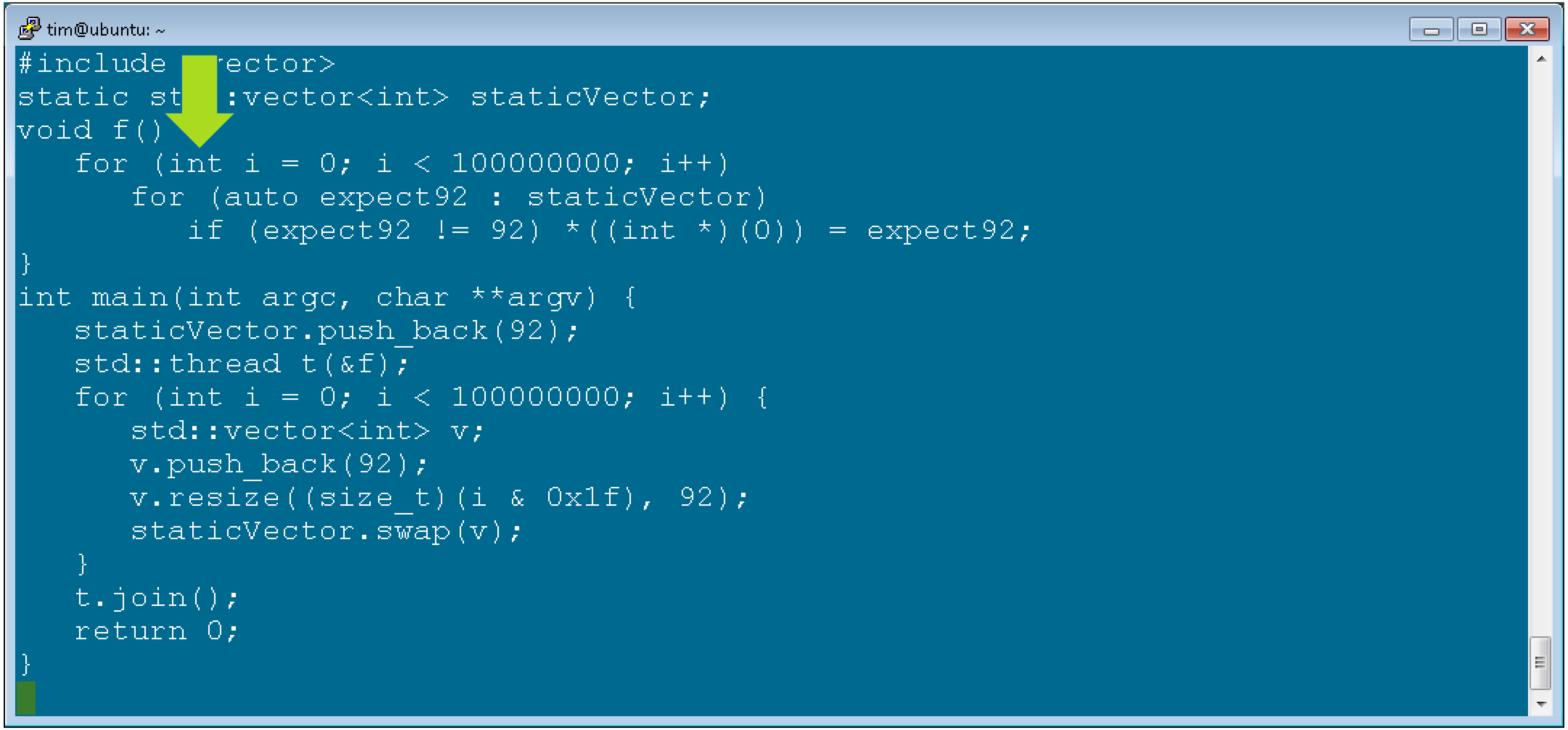
Using CHAP to Help With Crash Analysis

Analyze Corruption Issues – a Simulation



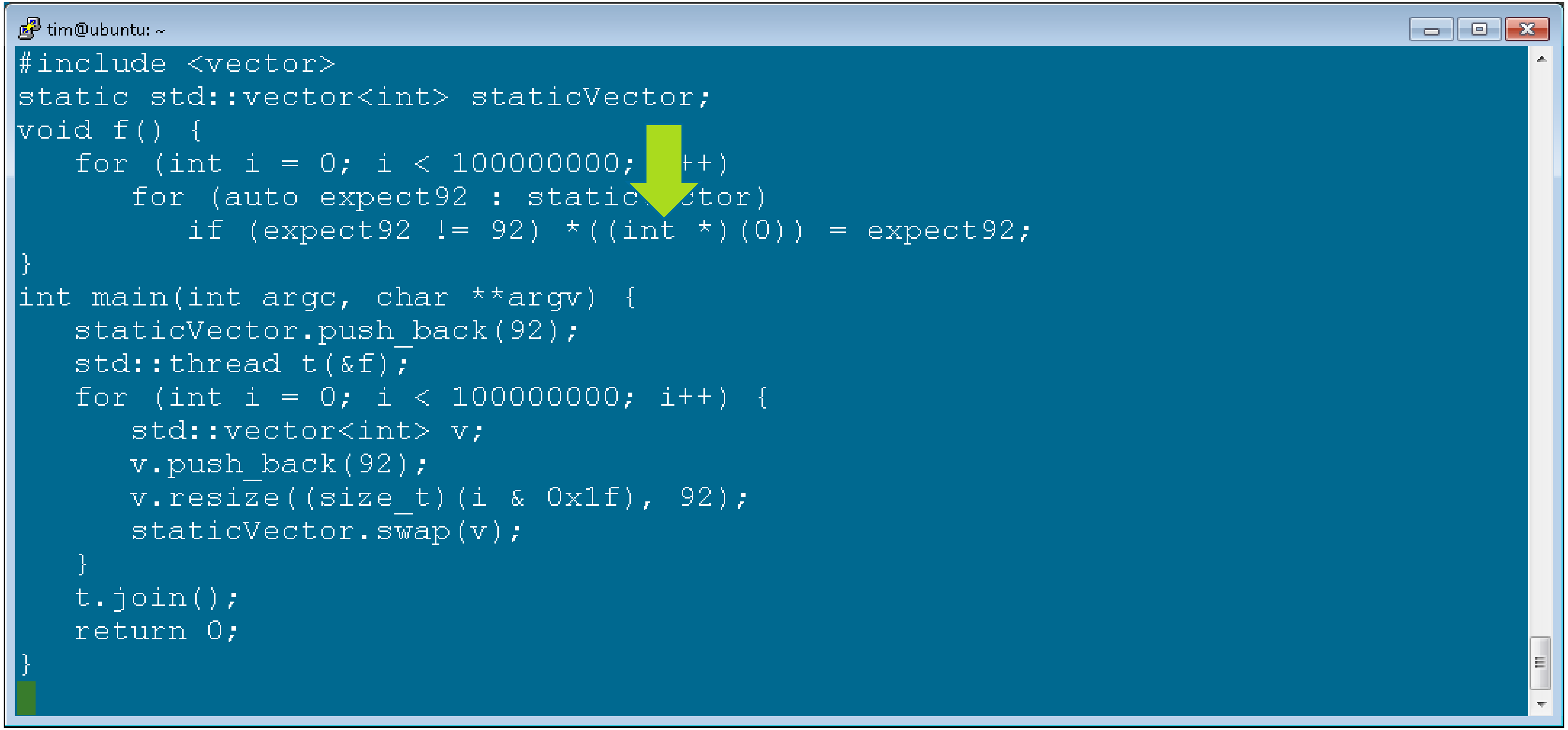
```
#include <vector>
static std::vector<int> staticVector;
void f() {
   for (int i = 0; i < 100000000; i++)
      for (auto expect92 : staticVector)
         if (expect92 != 92) *((int *)(0)) = expect92;
}
int main(int argc, char **argv) {
   staticVector.push_back(92);
   std::thread t(&f);
   for (int i = 0; i < 100000000; i++) {
      std::vector<int> v;
      v.push_back(92);
      v.resize((size_t)(i & 0x1f), 92);
      staticVector.swap(v);
   }
   t.join();
   return 0;
}
```

Analyze Corruption Issues – a Simulation



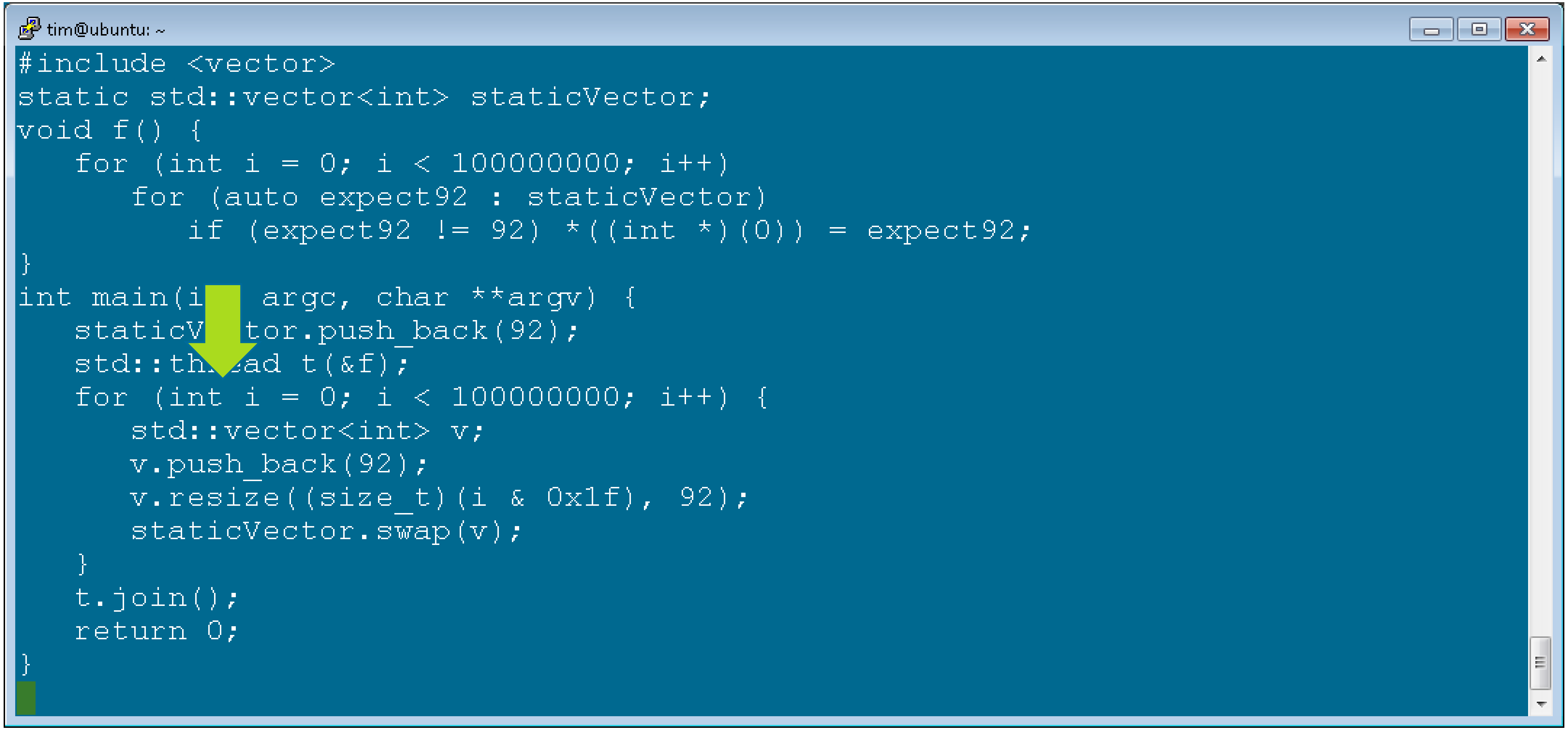
```
#include <vector>
static std::vector<int> staticVector;
void f() {
   for (int i = 0; i < 100000000; i++)
      for (auto expect92 : staticVector)
         if (expect92 != 92) *((int *)(0)) = expect92;
}
int main(int argc, char **argv) {
   staticVector.push_back(92);
   std::thread t(&f);
   for (int i = 0; i < 100000000; i++) {
      std::vector<int> v;
      v.push_back(92);
      v.resize((size_t)(i & 0x1f), 92);
      staticVector.swap(v);
   }
   t.join();
   return 0;
}
```

Analyze Corruption Issues – a Simulation



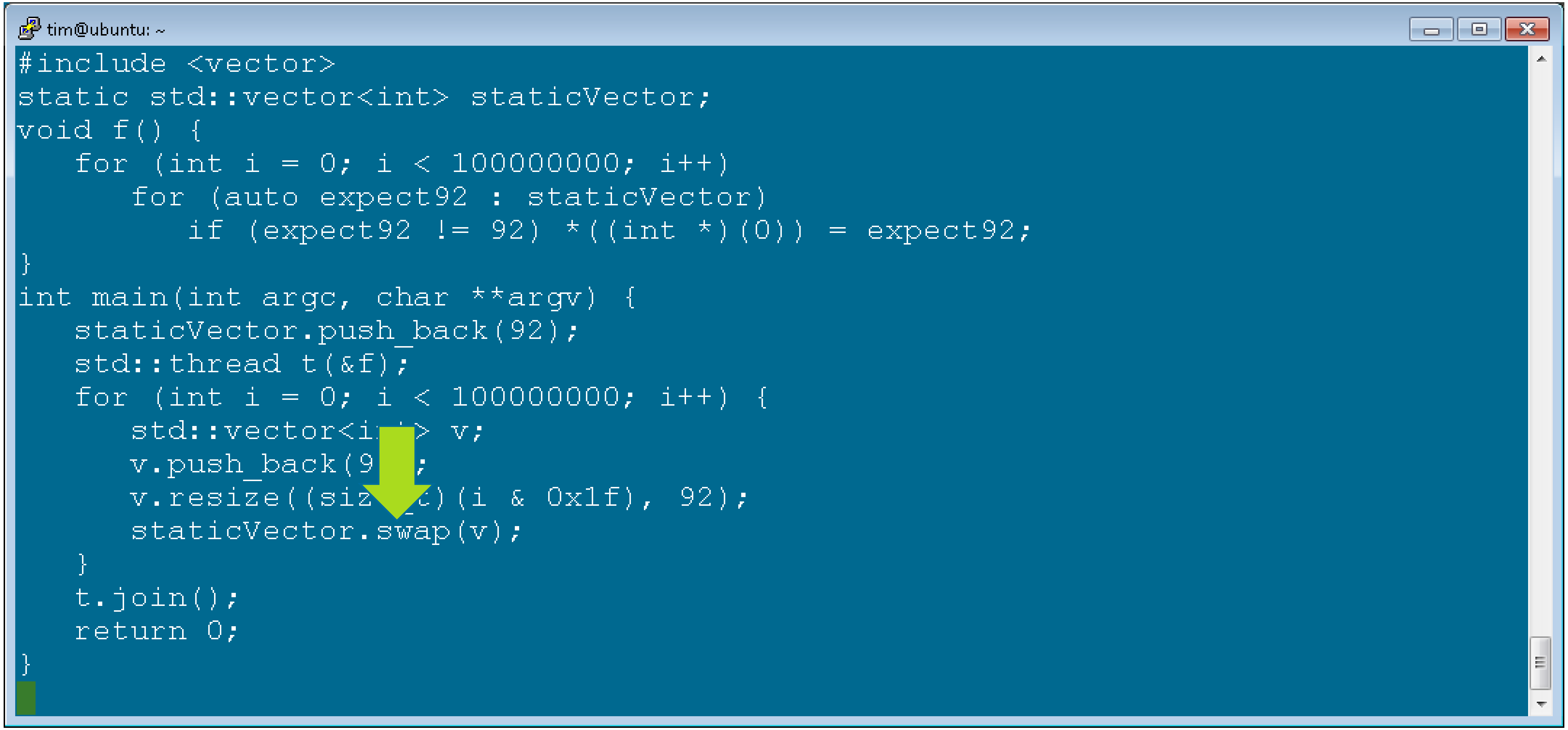
```
#include <vector>
static std::vector<int> staticVector;
void f() {
   for (int i = 0; i < 100000000; i++)
      for (auto expect92 : staticVector)
         if (expect92 != 92) *((int *)(0)) = expect92;
}
int main(int argc, char **argv) {
   staticVector.push_back(92);
   std::thread t(&f);
   for (int i = 0; i < 100000000; i++) {
      std::vector<int> v;
      v.push_back(92);
      v.resize((size_t)(i & 0x1f), 92);
      staticVector.swap(v);
   }
   t.join();
   return 0;
}
```

Analyze Corruption Issues – a Simulation



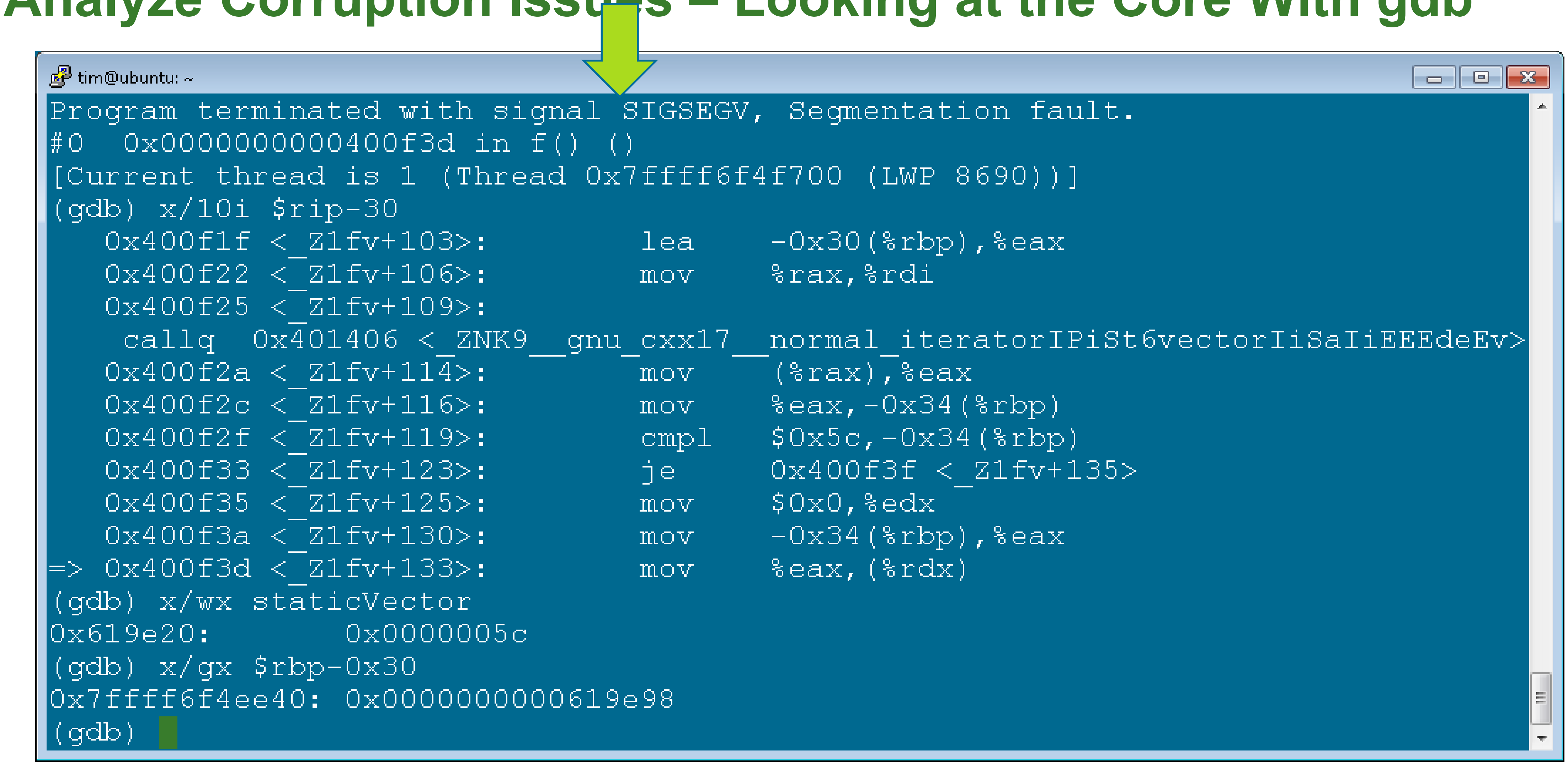
```
#include <vector>
static std::vector<int> staticVector;
void f() {
   for (int i = 0; i < 100000000; i++)
      for (auto expect92 : staticVector)
         if (expect92 != 92) *((int *)(0)) = expect92;
}
int main(int argc, char **argv) {
   staticVector.push_back(92);
   std::thread t(&f);
   for (int i = 0; i < 100000000; i++) {
      std::vector<int> v;
      v.push_back(92);
      v.resize((size_t)(i & 0x1f), 92);
      staticVector.swap(v);
   }
   t.join();
   return 0;
}
```

Analyze Corruption Issues – a Simulation



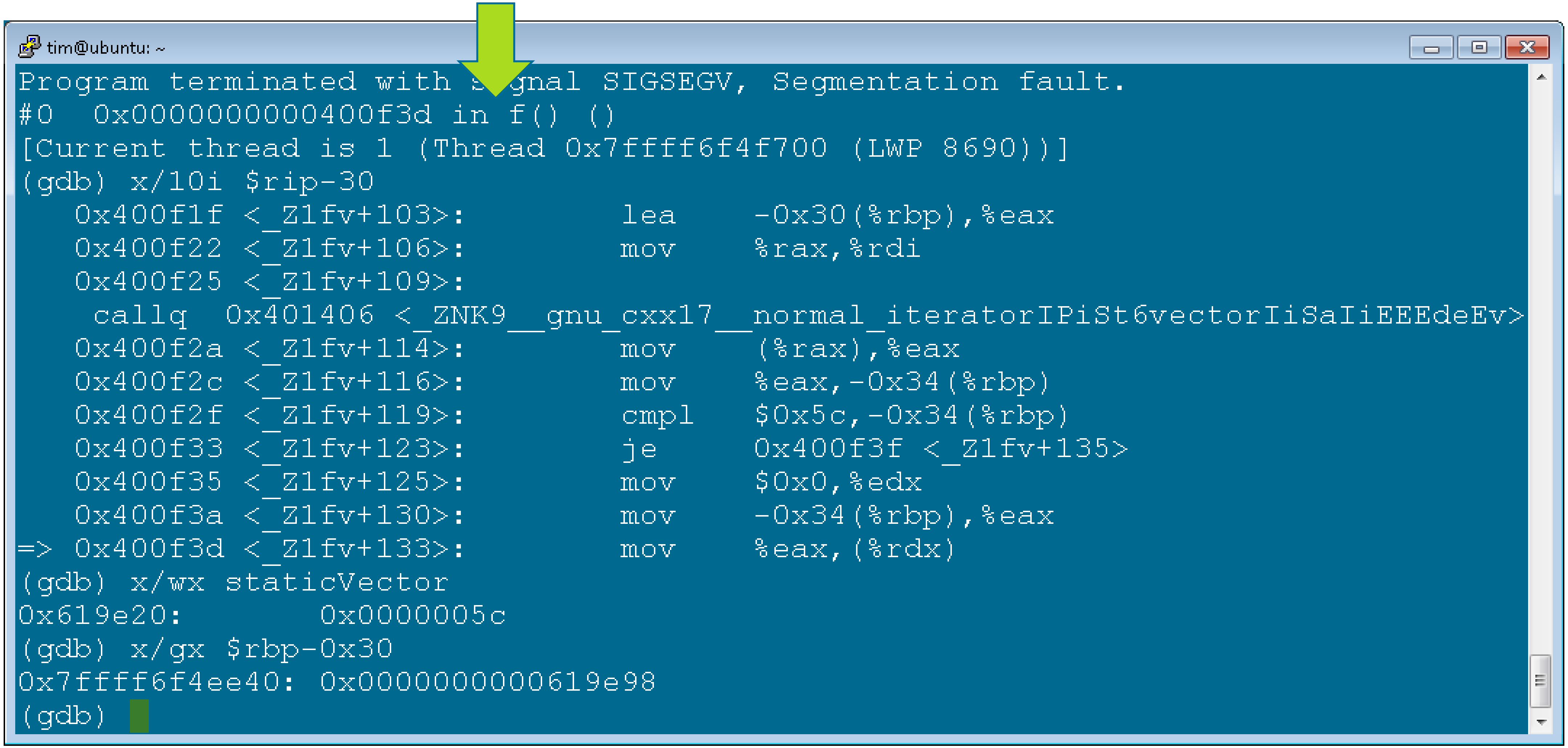
```
#include <vector>
static std::vector<int> staticVector;
void f() {
   for (int i = 0; i < 100000000; i++)
      for (auto expect92 : staticVector)
         if (expect92 != 92) *((int *)(0)) = expect92;
}
int main(int argc, char **argv) {
   staticVector.push_back(92);
   std::thread t(&f);
   for (int i = 0; i < 100000000; i++) {
      std::vector<int> v;
      v.push_back(92);
      v.resize((size_t)(i & 0x1f), 92);
      staticVector.swap(v);
   }
   t.join();
   return 0;
}
```

Analyze Corruption Issues – Looking at the Core With gdb



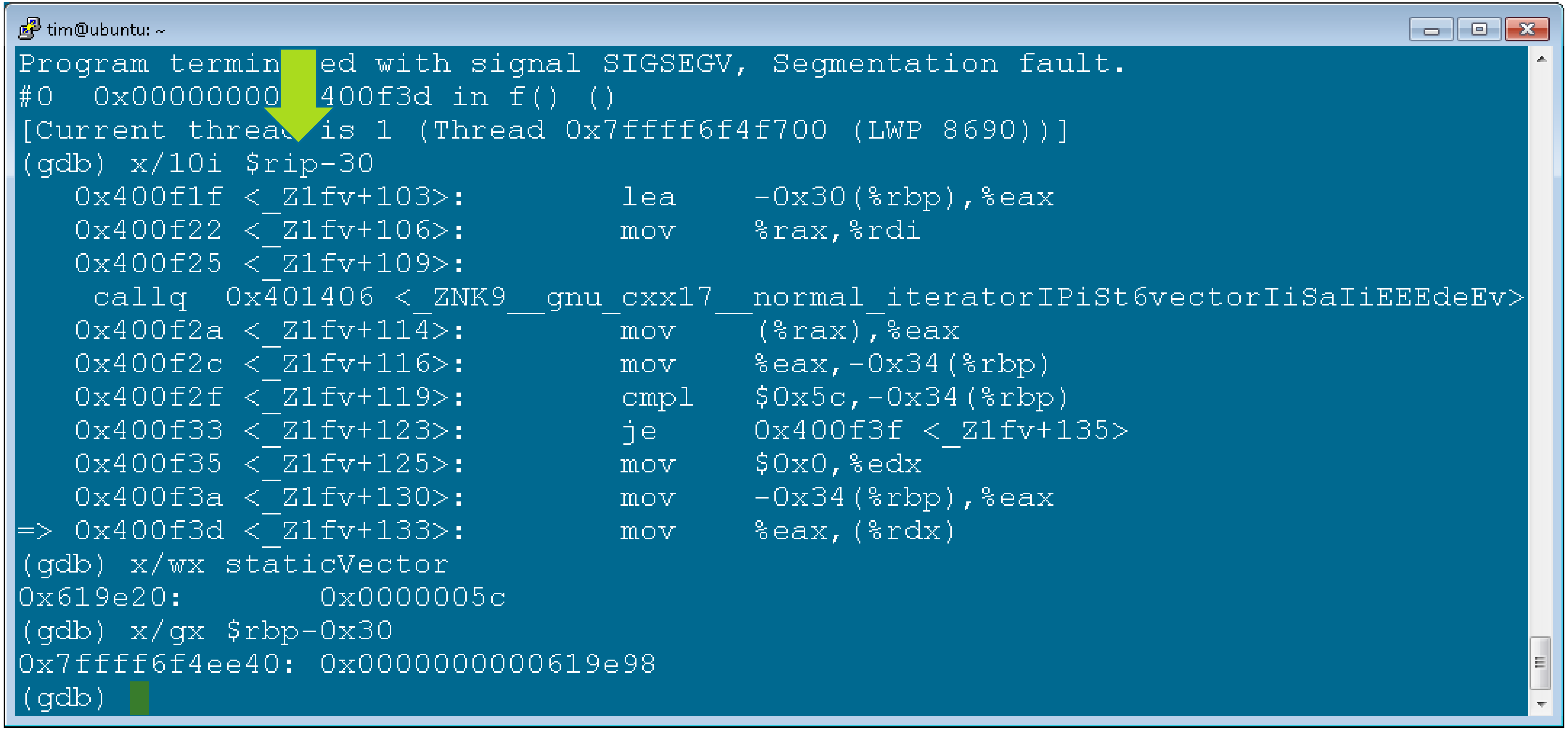
```
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x0000000000400f3d in f() ()
[Current thread is 1 (Thread 0x7ffff6f4f700 (LWP 8690))]
(gdb) x/10i $rip-30
   0x400f1f <_Z1fv+103>:        lea     -0x30(%rbp),%eax
   0x400f22 <_Z1fv+106>:        mov     %rax,%rdi
   0x400f25 <_Z1fv+109>:
    callq  0x401406 <_ZNK9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEdeEv>
   0x400f2a <_Z1fv+114>:        mov     (%rax),%eax
   0x400f2c <_Z1fv+116>:        mov     %eax,-0x34(%rbp)
   0x400f2f <_Z1fv+119>:        cmpl    $0x5c,-0x34(%rbp)
   0x400f33 <_Z1fv+123>:        je      0x400f3f <_Z1fv+135>
   0x400f35 <_Z1fv+125>:        mov     $0x0,%edx
   0x400f3a <_Z1fv+130>:        mov     -0x34(%rbp),%eax
=> 0x400f3d <_Z1fv+133>:        mov     %eax,(%rdx)
(gdb) x/wx staticVector
0x619e20:       0x0000005c
(gdb) x/gx $rbp-0x30
0x7ffff6f4ee40: 0x0000000000619e98
(gdb)
```

Analyze Corruption Issues – Looking at the Core With gdb



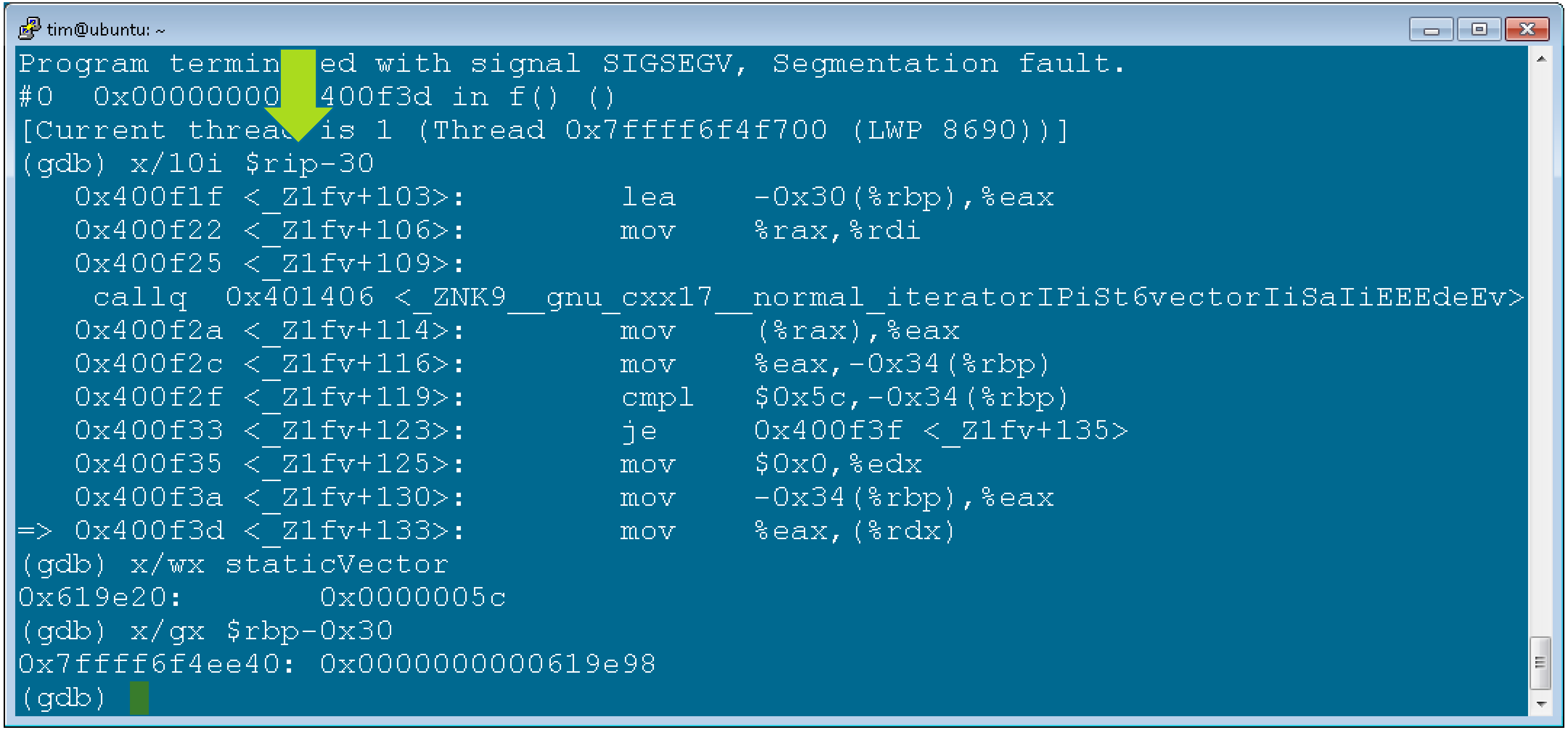
```
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x0000000000400f3d in f() ()
[Current thread is 1 (Thread 0x7ffff6f4f700 (LWP 8690))]
(gdb) x/10i $rip-30
   0x400f1f <_Z1fv+103>:        lea    -0x30(%rbp),%eax
   0x400f22 <_Z1fv+106>:        mov    %rax,%rdi
   0x400f25 <_Z1fv+109>:
    callq  0x401406 <_ZNK9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEdeEv>
   0x400f2a <_Z1fv+114>:        mov    (%rax),%eax
   0x400f2c <_Z1fv+116>:        mov    %eax,-0x34(%rbp)
   0x400f2f <_Z1fv+119>:        cmpl   $0x5c,-0x34(%rbp)
   0x400f33 <_Z1fv+123>:        je     0x400f3f <_Z1fv+135>
   0x400f35 <_Z1fv+125>:        mov    $0x0,%edx
   0x400f3a <_Z1fv+130>:        mov    -0x34(%rbp),%eax
=> 0x400f3d <_Z1fv+133>:        mov    %eax,(%rdx)
(gdb) x/wx staticVector
0x619e20:       0x0000005c
(gdb) x/gx $rbp-0x30
0x7ffff6f4ee40: 0x0000000000619e98
(gdb)
```

Analyze Corruption Issues – Looking at the Core With gdb



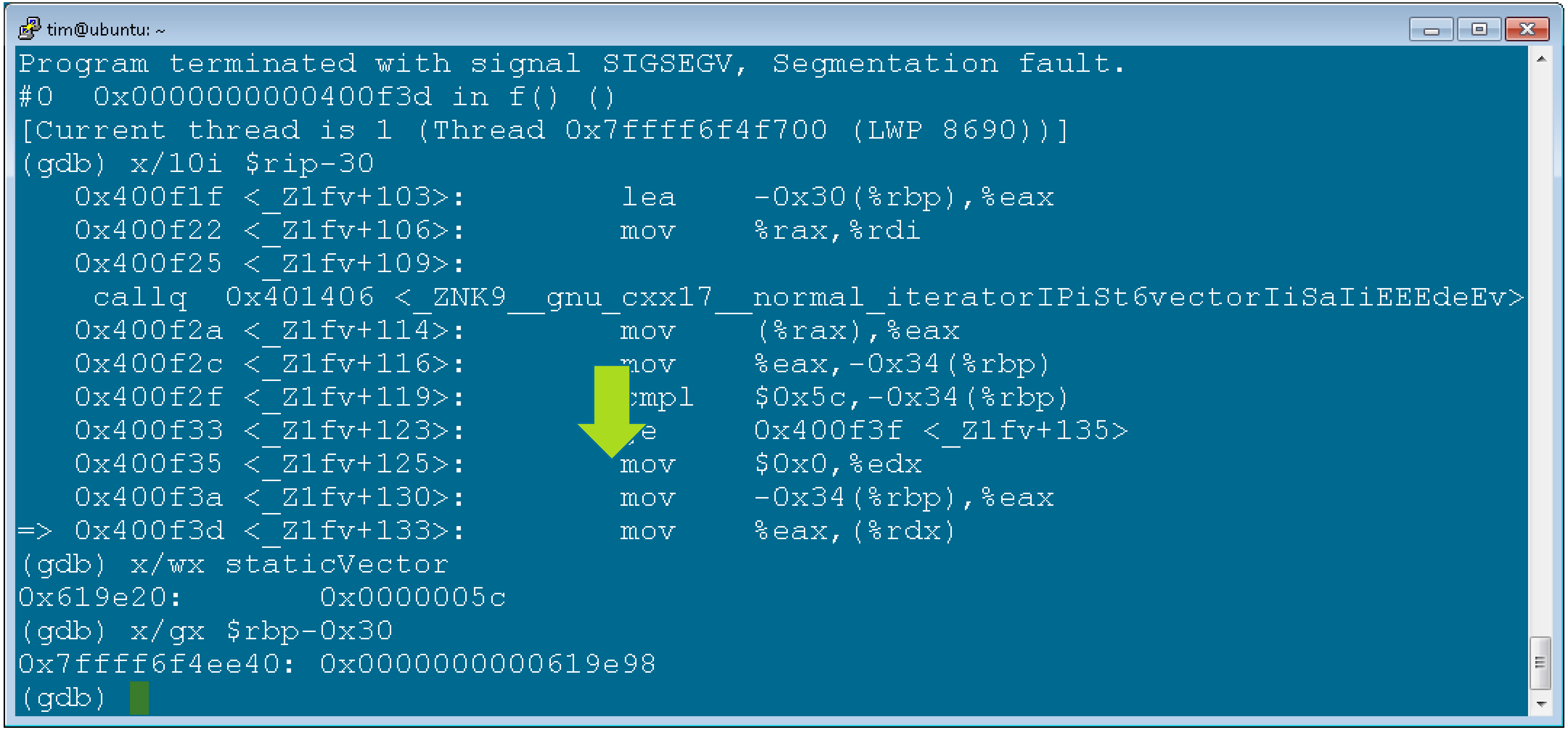
```
Program termin  ed with signal SIGSEGV, Segmentation fault.
#0  0x00000000  400f3d in f() ()
[Current threa  is 1 (Thread 0x7ffff6f4f700 (LWP 8690))]
(gdb) x/10i $rip-30
   0x400f1f <_Z1fv+103>:        lea    -0x30(%rbp),%eax
   0x400f22 <_Z1fv+106>:        mov    %rax,%rdi
   0x400f25 <_Z1fv+109>:
    callq  0x401406 <_ZNK9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEdeEv>
   0x400f2a <_Z1fv+114>:        mov    (%rax),%eax
   0x400f2c <_Z1fv+116>:        mov    %eax,-0x34(%rbp)
   0x400f2f <_Z1fv+119>:        cmpl   $0x5c,-0x34(%rbp)
   0x400f33 <_Z1fv+123>:        je     0x400f3f <_Z1fv+135>
   0x400f35 <_Z1fv+125>:        mov    $0x0,%edx
   0x400f3a <_Z1fv+130>:        mov    -0x34(%rbp),%eax
=> 0x400f3d <_Z1fv+133>:        mov    %eax,(%rdx)
(gdb) x/wx staticVector
0x619e20:       0x0000005c
(gdb) x/gx $rbp-0x30
0x7ffff6f4ee40: 0x0000000000619e98
(gdb)
```

Analyze Corruption Issues – Looking at the Core With gdb



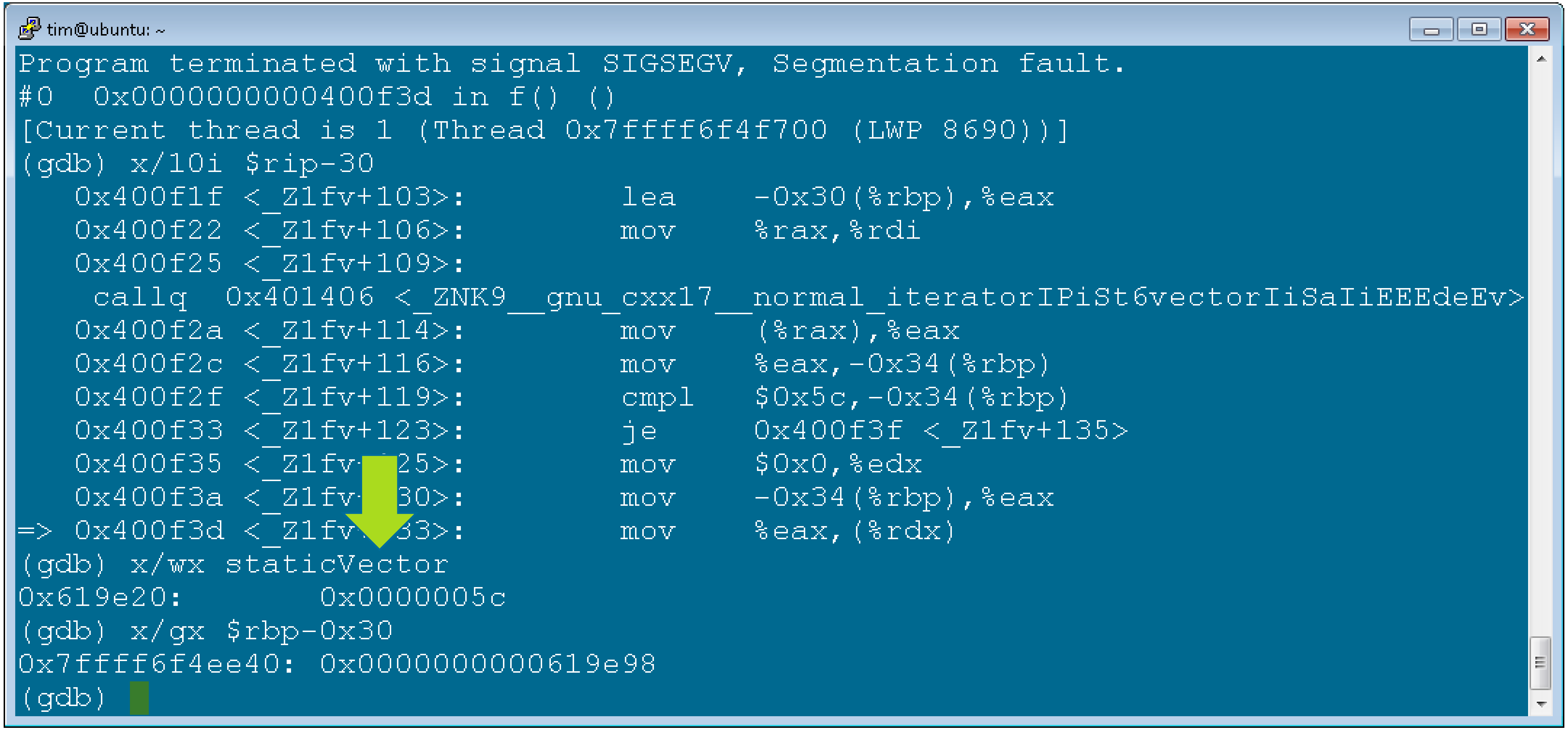
```
Program termin  ed with signal SIGSEGV, Segmentation fault.
#0  0x00000000  400f3d in f() ()
[Current threa  is 1 (Thread 0x7ffff6f4f700 (LWP 8690))]
(gdb) x/10i $rip-30
   0x400f1f <_Z1fv+103>:        lea    -0x30(%rbp),%eax
   0x400f22 <_Z1fv+106>:        mov    %rax,%rdi
   0x400f25 <_Z1fv+109>:
    callq  0x401406 <_ZNK9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEdeEv>
   0x400f2a <_Z1fv+114>:        mov    (%rax),%eax
   0x400f2c <_Z1fv+116>:        mov    %eax,-0x34(%rbp)
   0x400f2f <_Z1fv+119>:        cmpl   $0x5c,-0x34(%rbp)
   0x400f33 <_Z1fv+123>:        je     0x400f3f <_Z1fv+135>
   0x400f35 <_Z1fv+125>:        mov    $0x0,%edx
   0x400f3a <_Z1fv+130>:        mov    -0x34(%rbp),%eax
=> 0x400f3d <_Z1fv+133>:        mov    %eax,(%rdx)
(gdb) x/wx staticVector
0x619e20:       0x0000005c
(gdb) x/gx $rbp-0x30
0x7ffff6f4ee40: 0x0000000000619e98
(gdb) 
```

Analyze Corruption Issues – Looking at the Core With gdb



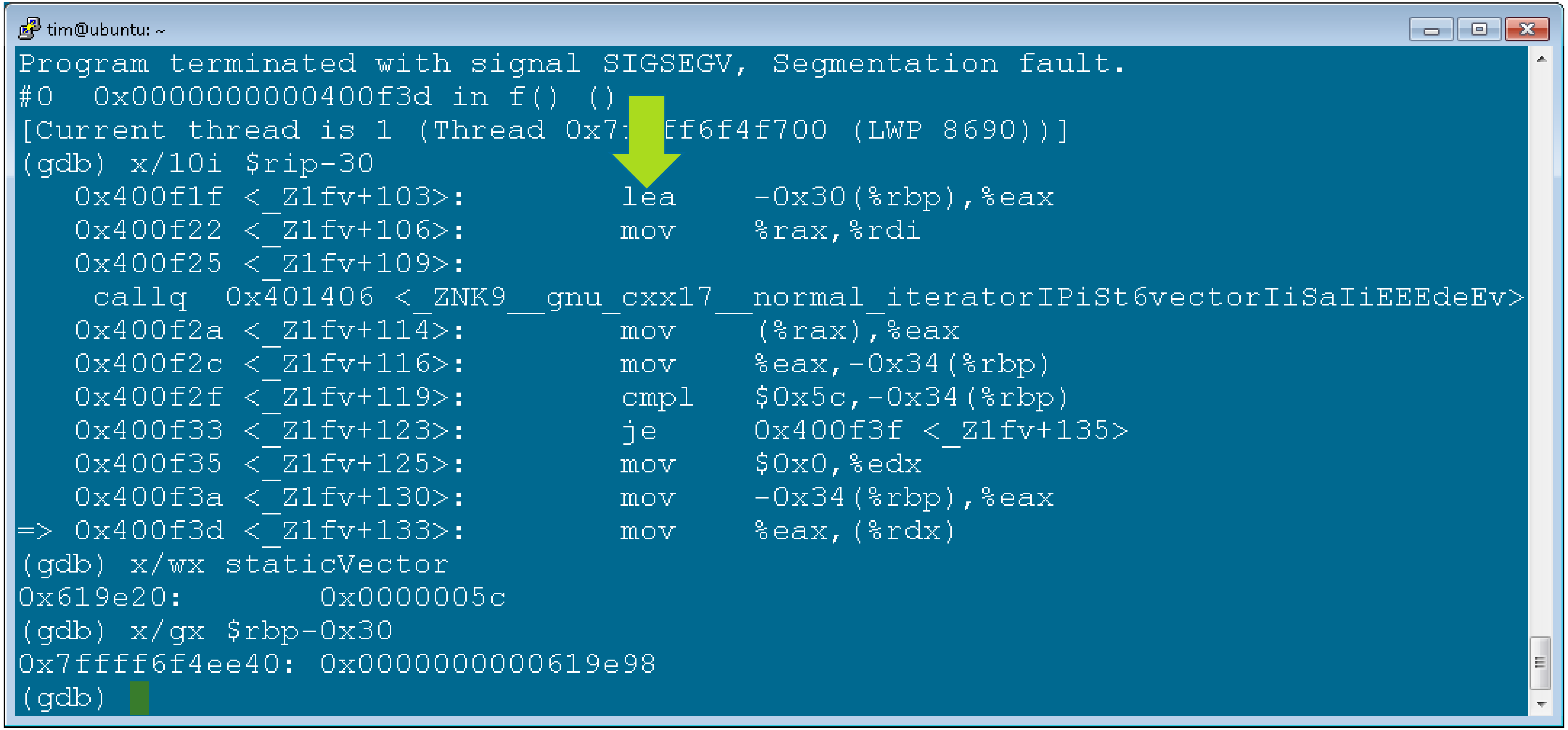
```
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x0000000000400f3d in f() ()
[Current thread is 1 (Thread 0x7ffff6f4f700 (LWP 8690))]
(gdb) x/10i $rip-30
   0x400f1f <_Z1fv+103>:        lea    -0x30(%rbp),%eax
   0x400f22 <_Z1fv+106>:        mov    %rax,%rdi
   0x400f25 <_Z1fv+109>:
    callq  0x401406 <_ZNK9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEdeEv>
   0x400f2a <_Z1fv+114>:        mov    (%rax),%eax
   0x400f2c <_Z1fv+116>:        mov    %eax,-0x34(%rbp)
   0x400f2f <_Z1fv+119>:        cmpl   $0x5c,-0x34(%rbp)
   0x400f33 <_Z1fv+123>:        je     0x400f3f <_Z1fv+135>
   0x400f35 <_Z1fv+125>:        mov    $0x0,%edx
   0x400f3a <_Z1fv+130>:        mov    -0x34(%rbp),%eax
=> 0x400f3d <_Z1fv+133>:        mov    %eax,(%rdx)
(gdb) x/wx staticVector
0x619e20:       0x0000005c
(gdb) x/gx $rbp-0x30
0x7ffff6f4ee40: 0x0000000000619e98
(gdb)
```

Analyze Corruption Issues – Looking at the Core With gdb



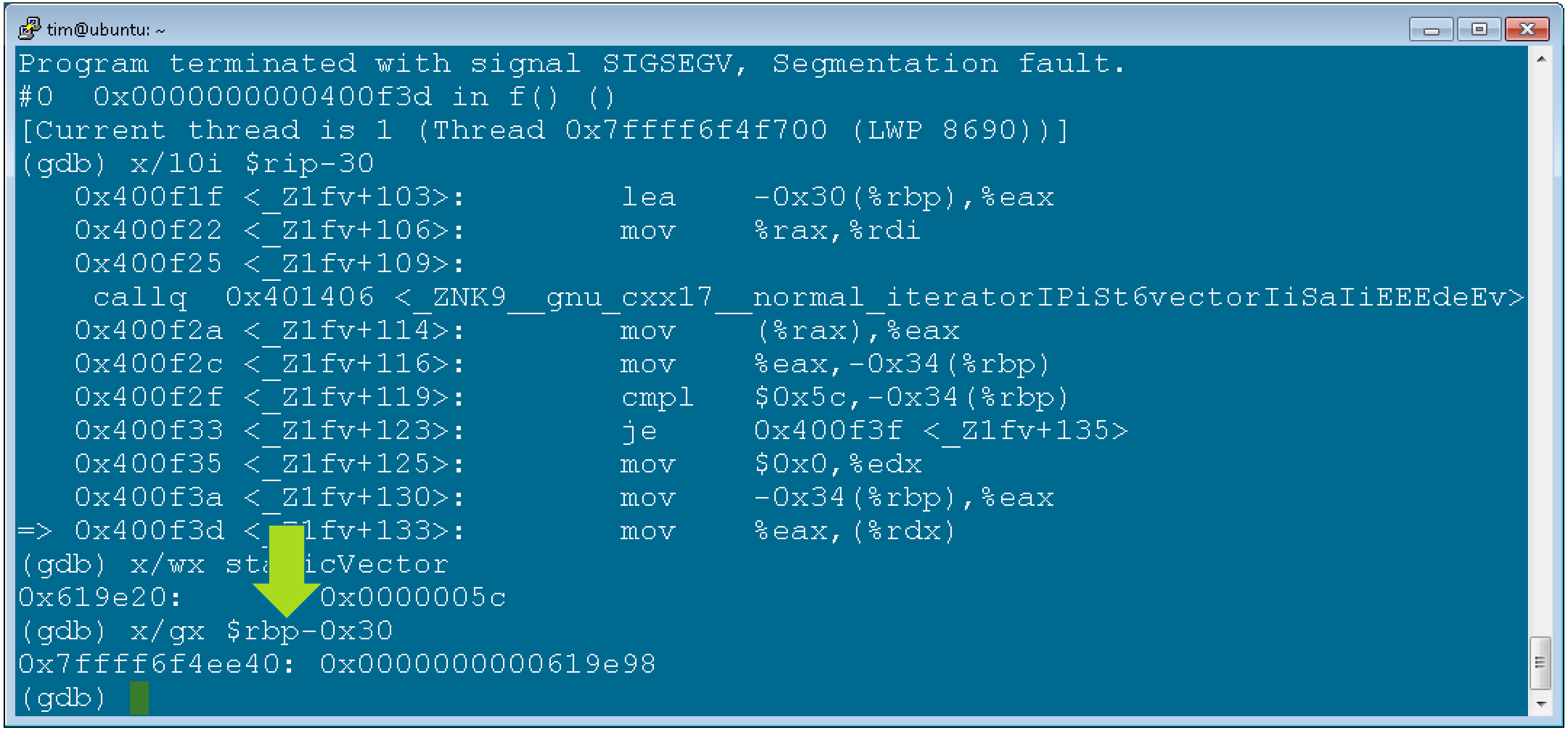
```
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x0000000000400f3d in f() ()
[Current thread is 1 (Thread 0x7ffff6f4f700 (LWP 8690))]
(gdb) x/10i $rip-30
   0x400f1f <_Z1fv+103>:        lea    -0x30(%rbp),%eax
   0x400f22 <_Z1fv+106>:        mov    %rax,%rdi
   0x400f25 <_Z1fv+109>:
    callq  0x401406 <_ZNK9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEdeEv>
   0x400f2a <_Z1fv+114>:        mov    (%rax),%eax
   0x400f2c <_Z1fv+116>:        mov    %eax,-0x34(%rbp)
   0x400f2f <_Z1fv+119>:        cmpl   $0x5c,-0x34(%rbp)
   0x400f33 <_Z1fv+123>:        je     0x400f3f <_Z1fv+135>
   0x400f35 <_Z1fv+125>:        mov    $0x0,%edx
   0x400f3a <_Z1fv+130>:        mov    -0x34(%rbp),%eax
=> 0x400f3d <_Z1fv+133>:        mov    %eax,(%rdx)
(gdb) x/wx staticVector
0x619e20:       0x0000005c
(gdb) x/gx $rbp-0x30
0x7ffff6f4ee40: 0x0000000000619e98
(gdb)
```

Analyze Corruption Issues – Looking at the Core With gdb



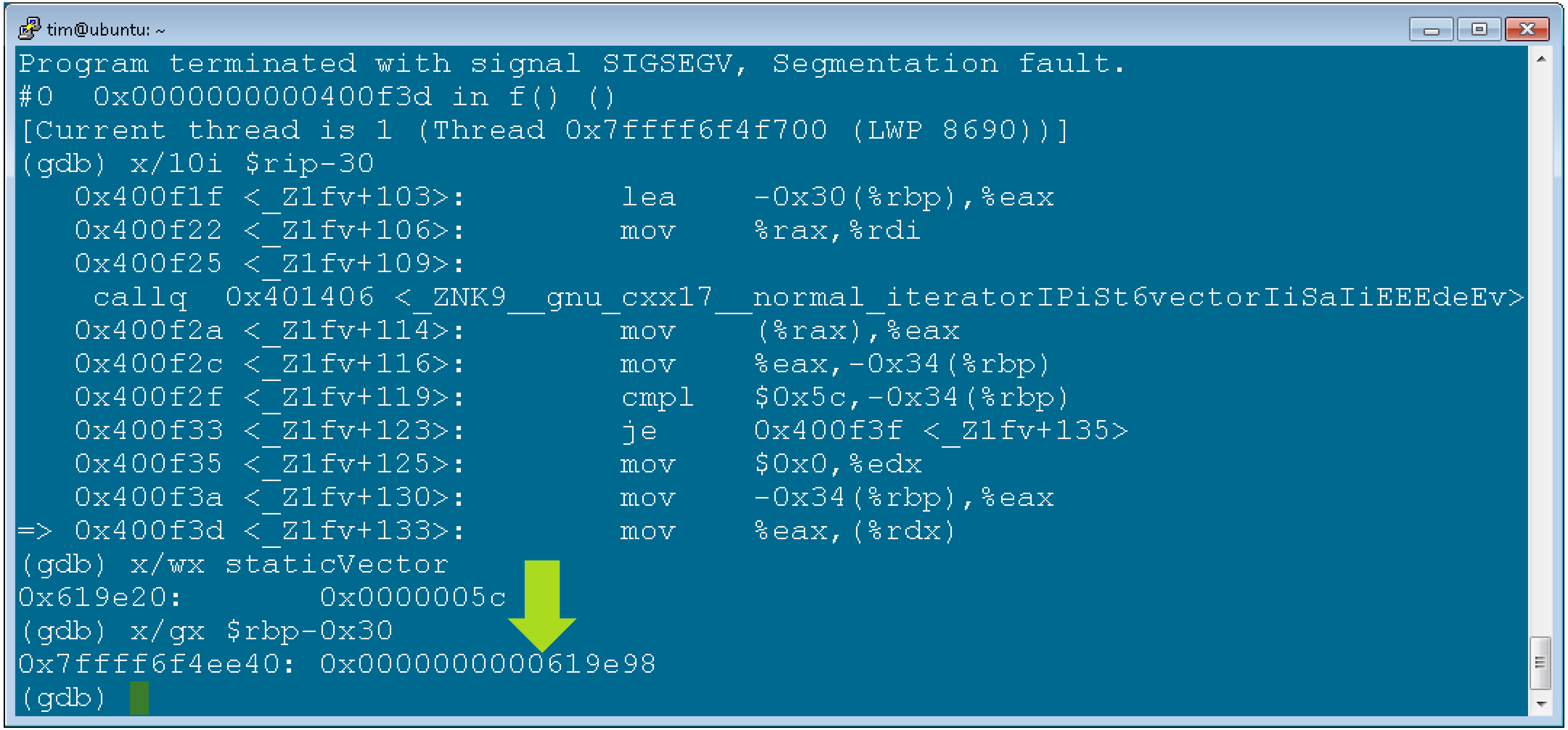
```
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x0000000000400f3d in f() ()
[Current thread is 1 (Thread 0x7ffff6f4f700 (LWP 8690))]
(gdb) x/10i $rip-30
   0x400f1f <_Z1fv+103>:        lea    -0x30(%rbp),%eax
   0x400f22 <_Z1fv+106>:        mov    %rax,%rdi
   0x400f25 <_Z1fv+109>:
    callq  0x401406 <_ZNK9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEdeEv>
   0x400f2a <_Z1fv+114>:        mov    (%rax),%eax
   0x400f2c <_Z1fv+116>:        mov    %eax,-0x34(%rbp)
   0x400f2f <_Z1fv+119>:        cmpl   $0x5c,-0x34(%rbp)
   0x400f33 <_Z1fv+123>:        je     0x400f3f <_Z1fv+135>
   0x400f35 <_Z1fv+125>:        mov    $0x0,%edx
   0x400f3a <_Z1fv+130>:        mov    -0x34(%rbp),%eax
=> 0x400f3d <_Z1fv+133>:        mov    %eax,(%rdx)
(gdb) x/wx staticVector
0x619e20:       0x0000005c
(gdb) x/gx $rbp-0x30
0x7ffff6f4ee40: 0x0000000000619e98
(gdb)
```

Analyze Corruption Issues – Looking at the Core With gdb



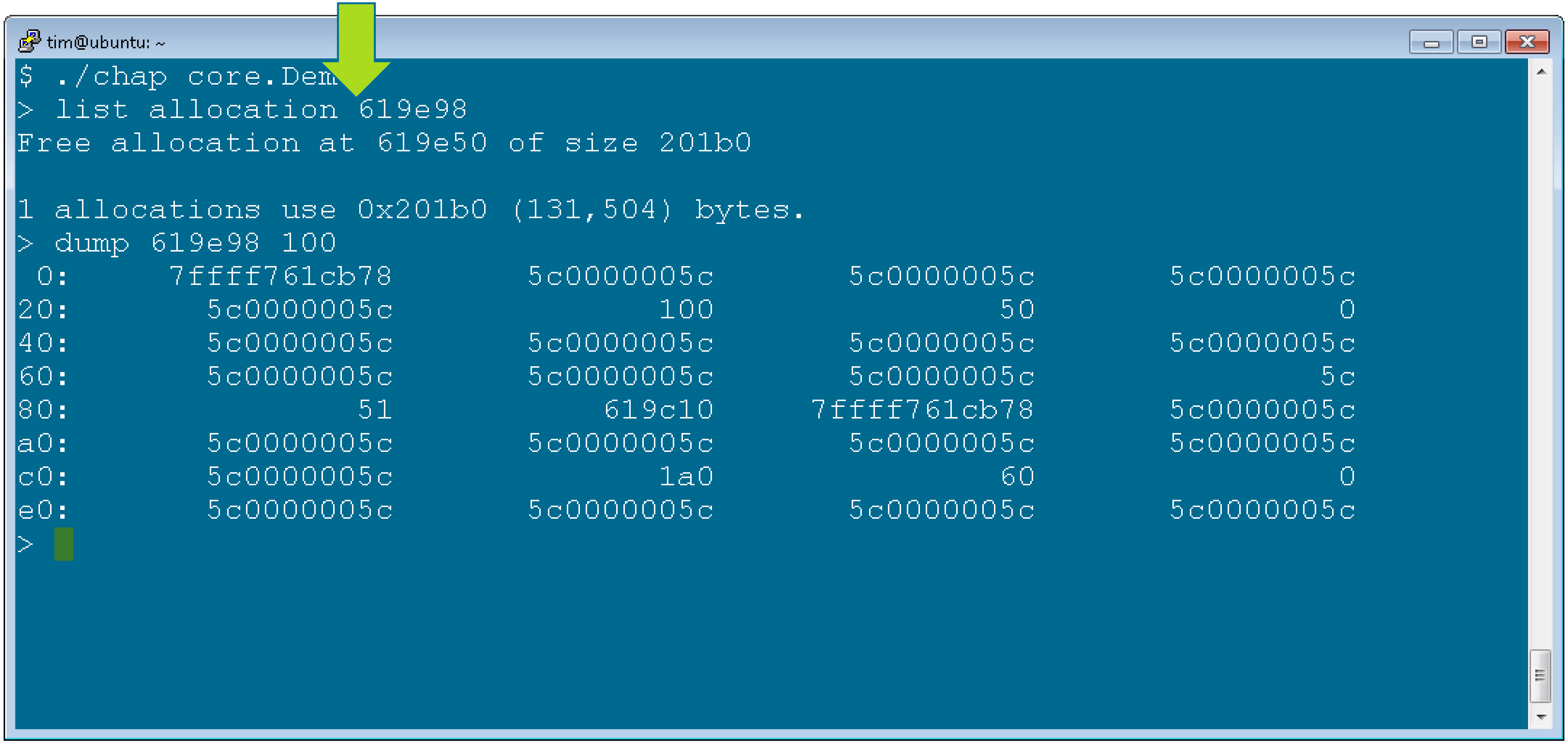
```
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x0000000000400f3d in f() ()
[Current thread is 1 (Thread 0x7ffff6f4f700 (LWP 8690))]
(gdb) x/10i $rip-30
   0x400f1f <_Z1fv+103>:        lea    -0x30(%rbp),%eax
   0x400f22 <_Z1fv+106>:        mov    %rax,%rdi
   0x400f25 <_Z1fv+109>:
    callq  0x401406 <_ZNK9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEdeEv>
   0x400f2a <_Z1fv+114>:        mov    (%rax),%eax
   0x400f2c <_Z1fv+116>:        mov    %eax,-0x34(%rbp)
   0x400f2f <_Z1fv+119>:        cmpl   $0x5c,-0x34(%rbp)
   0x400f33 <_Z1fv+123>:        je     0x400f3f <_Z1fv+135>
   0x400f35 <_Z1fv+125>:        mov    $0x0,%edx
   0x400f3a <_Z1fv+130>:        mov    -0x34(%rbp),%eax
=> 0x400f3d <_Z1fv+133>:        mov    %eax,(%rdx)
(gdb) x/wx staticVector
0x619e20:       0x0000005c
(gdb) x/gx $rbp-0x30
0x7ffff6f4ee40: 0x0000000000619e98
(gdb)
```

Analyze Corruption Issues – Looking at the Core With gdb



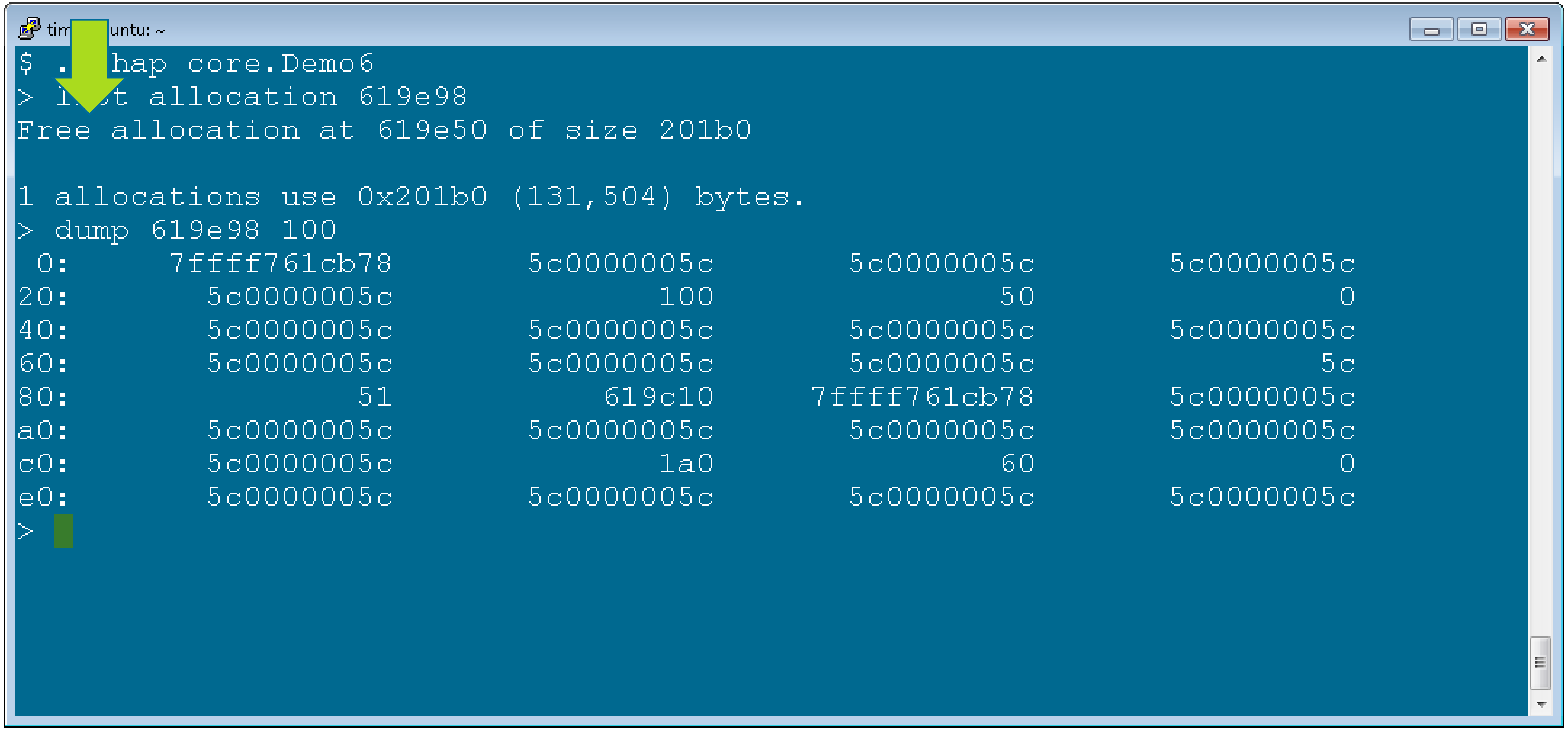
```
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x0000000000400f3d in f() ()
[Current thread is 1 (Thread 0x7ffff6f4f700 (LWP 8690))]
(gdb) x/10i $rip-30
   0x400f1f <_Z1fv+103>:        lea    -0x30(%rbp),%eax
   0x400f22 <_Z1fv+106>:        mov    %rax,%rdi
   0x400f25 <_Z1fv+109>:
    callq  0x401406 <_ZNK9__gnu_cxx17__normal_iteratorIPiSt6vectorIiSaIiEEEdeEv>
   0x400f2a <_Z1fv+114>:        mov    (%rax),%eax
   0x400f2c <_Z1fv+116>:        mov    %eax,-0x34(%rbp)
   0x400f2f <_Z1fv+119>:        cmpl   $0x5c,-0x34(%rbp)
   0x400f33 <_Z1fv+123>:        je     0x400f3f <_Z1fv+135>
   0x400f35 <_Z1fv+125>:        mov    $0x0,%edx
   0x400f3a <_Z1fv+130>:        mov    -0x34(%rbp),%eax
=> 0x400f3d <_Z1fv+133>:        mov    %eax,(%rdx)
(gdb) x/wx staticVector
0x619e20:       0x0000005c
(gdb) x/gx $rbp-0x30
0x7ffff6f4ee40: 0x0000000000619e98
(gdb)
```

Analyze Corruption Issues – Looking at the Core With CHAP



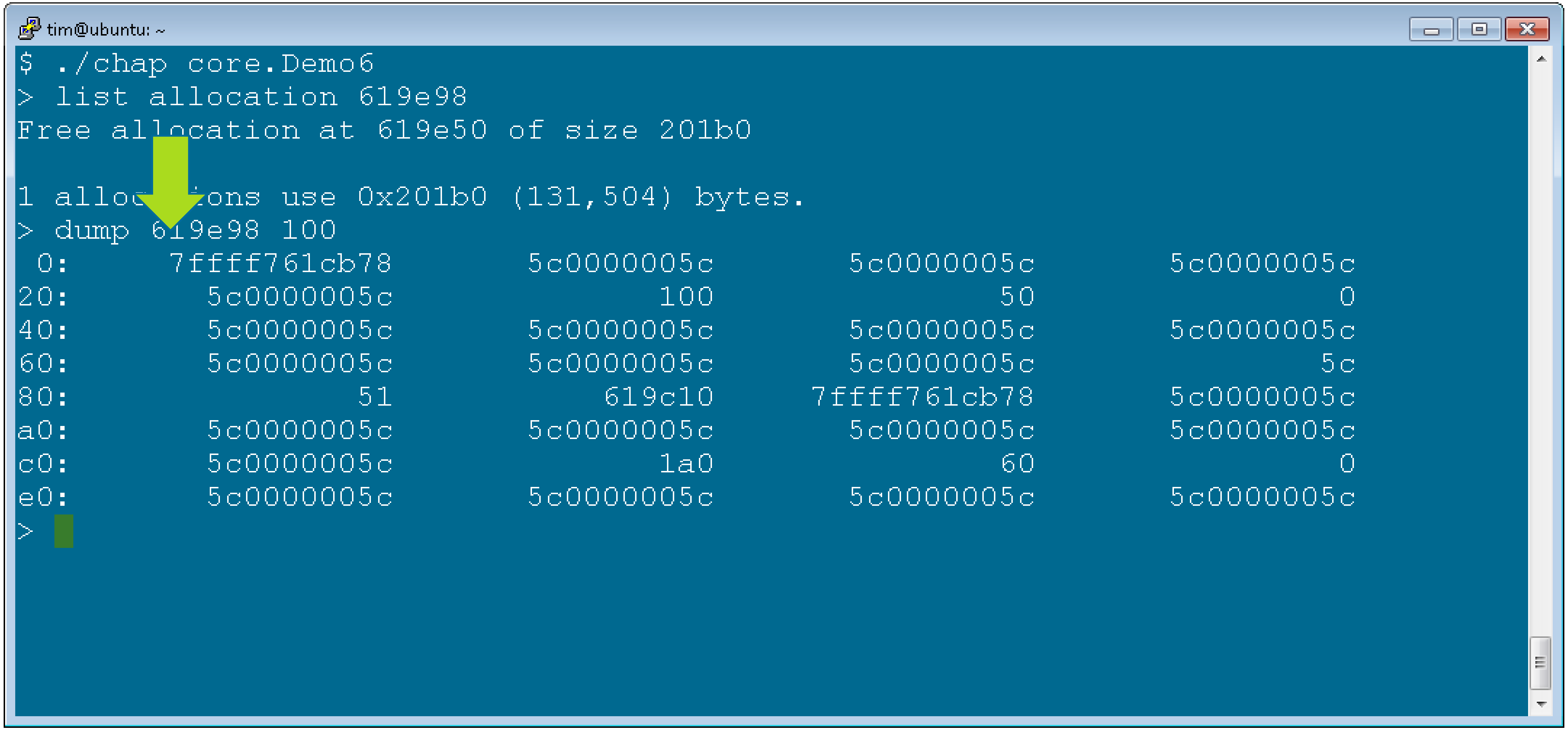
```
$ ./chap core.Dem
> list allocation 619e98
Free allocation at 619e50 of size 201b0

1 allocations use 0x201b0 (131,504) bytes.
> dump 619e98 100
 0:     7ffff761cb78        5c0000005c        5c0000005c        5c0000005c
20:       5c0000005c               100                50                 0
40:       5c0000005c        5c0000005c        5c0000005c        5c0000005c
60:       5c0000005c        5c0000005c        5c0000005c                5c
80:               51            619c10      7ffff761cb78        5c0000005c
a0:       5c0000005c        5c0000005c        5c0000005c        5c0000005c
c0:       5c0000005c               1a0                60                 0
e0:       5c0000005c        5c0000005c        5c0000005c        5c0000005c
>
```

Analyze Corruption Issues – Looking at the Core With CHAP



```
$ ./chap core.Demo6
> list allocation 619e98
Free allocation at 619e50 of size 201b0

1 allocations use 0x201b0 (131,504) bytes.
> dump 619e98 100
 0:     7ffff761cb78       5c0000005c       5c0000005c       5c0000005c
20:       5c0000005c              100               50                0
40:       5c0000005c       5c0000005c       5c0000005c       5c0000005c
60:       5c0000005c       5c0000005c       5c0000005c               5c
80:               51           619c10     7ffff761cb78       5c0000005c
a0:       5c0000005c       5c0000005c       5c0000005c       5c0000005c
c0:       5c0000005c              1a0               60                0
e0:       5c0000005c       5c0000005c       5c0000005c       5c0000005c
> 
```

Analyze Corruption Issues – Looking at the Core With CHAP

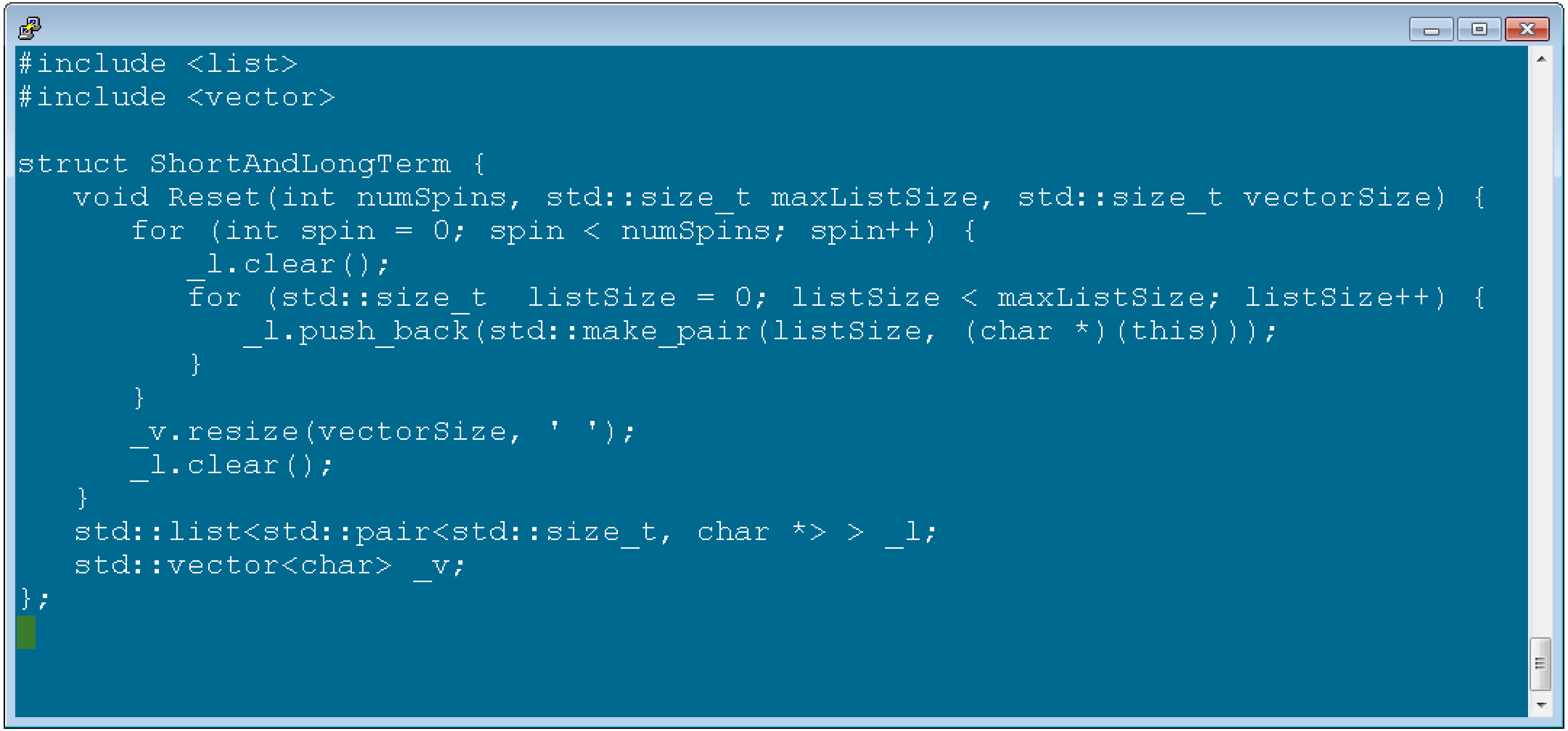


```
$ ./chap core.Demo6
> list allocation 619e98
Free allocation at 619e50 of size 201b0

1 allocations use 0x201b0 (131,504) bytes.
> dump 619e98 100
 0:      7ffff761cb78        5c0000005c        5c0000005c        5c0000005c
20:        5c0000005c               100                50                 0
40:        5c0000005c        5c0000005c        5c0000005c        5c0000005c
60:        5c0000005c        5c0000005c        5c0000005c                5c
80:                51            619c10      7ffff761cb78        5c0000005c
a0:        5c0000005c        5c0000005c        5c0000005c        5c0000005c
c0:        5c0000005c               1a0                60                 0
e0:        5c0000005c        5c0000005c        5c0000005c        5c0000005c
>
```

Using CHAP to Examine Overhead

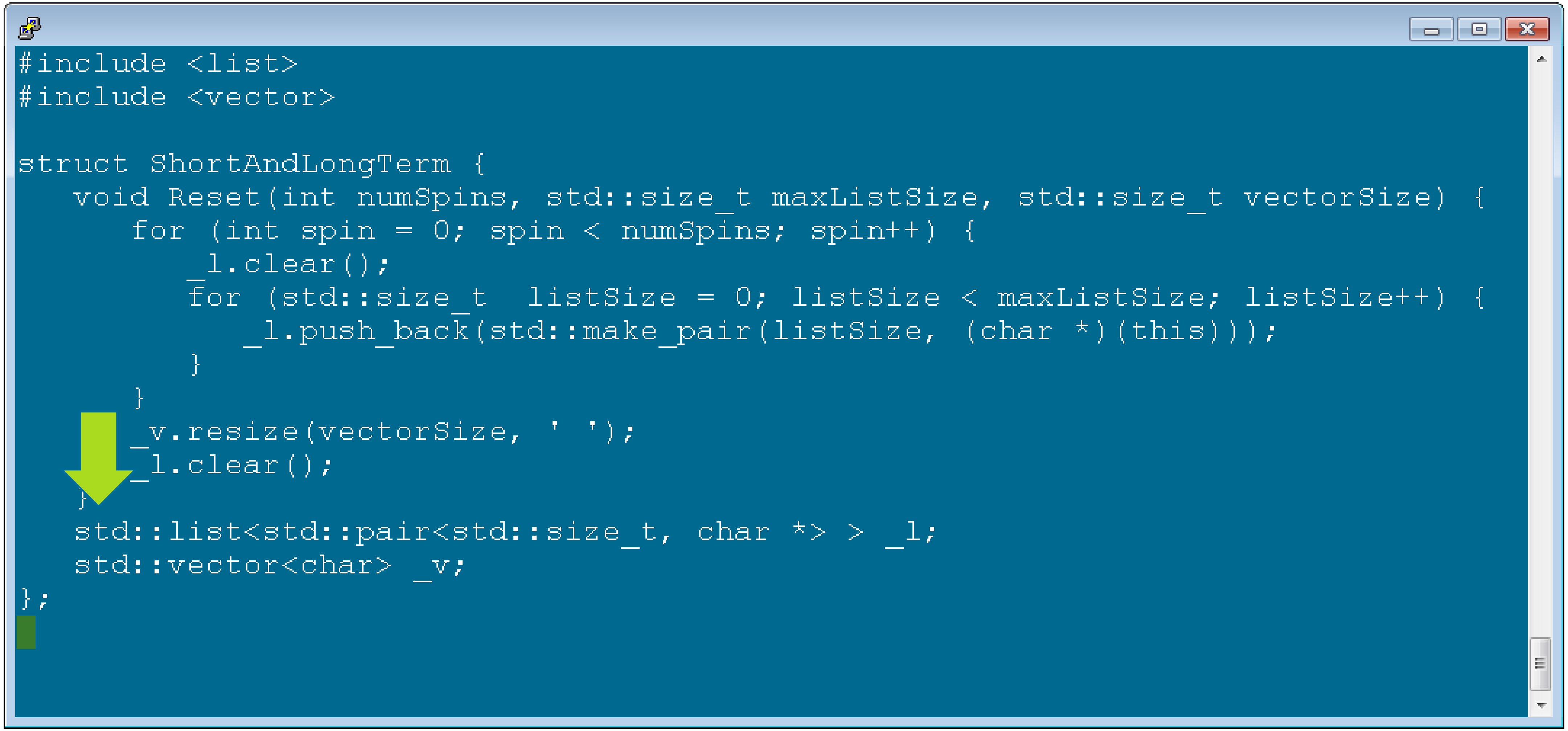
vmware®

Understanding Overhead: A Simulation Utility Class



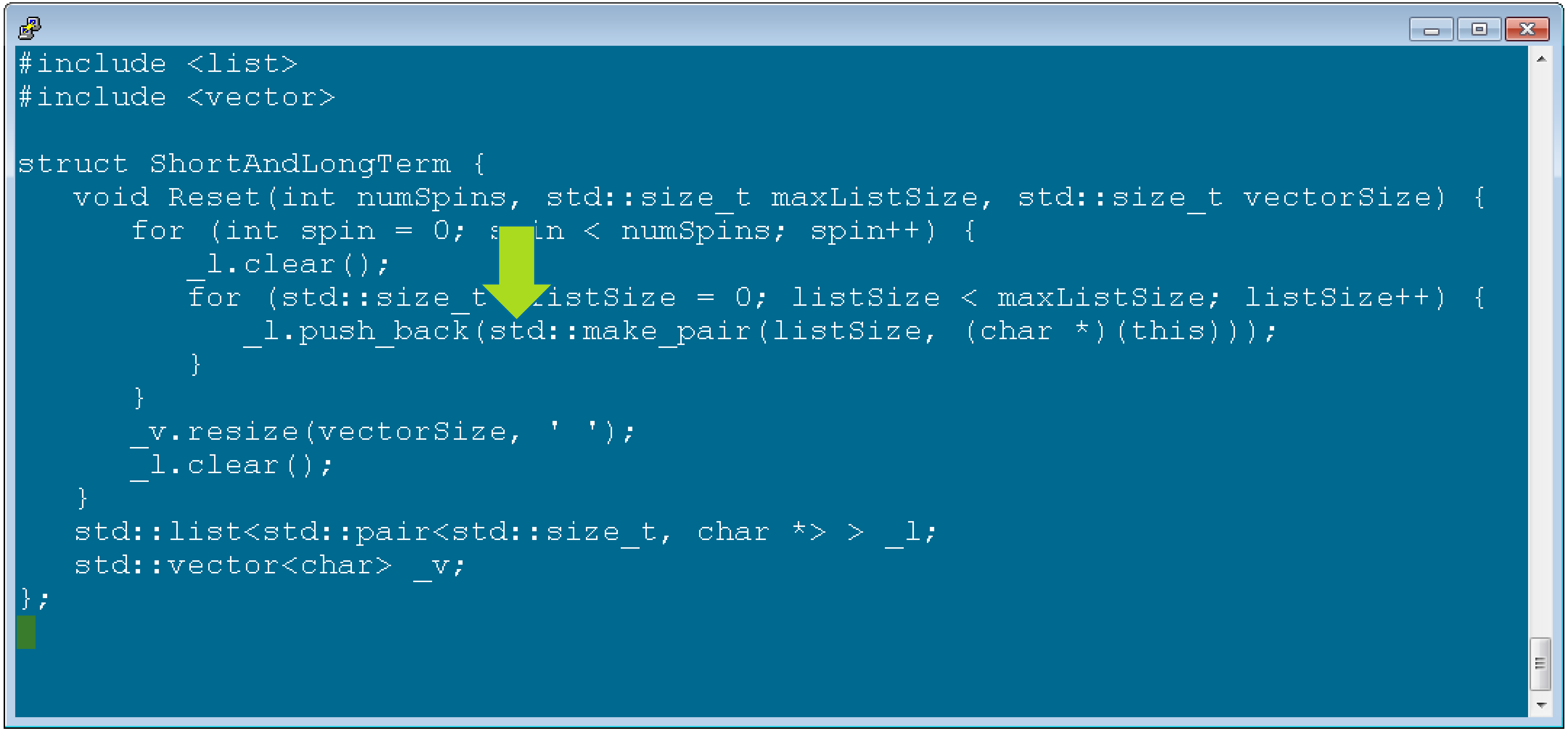
```
#include <list>
#include <vector>

struct ShortAndLongTerm {
   void Reset(int numSpins, std::size_t maxListSize, std::size_t vectorSize) {
      for (int spin = 0; spin < numSpins; spin++) {
         _l.clear();
         for (std::size_t  listSize = 0; listSize < maxListSize; listSize++) {
            _l.push_back(std::make_pair(listSize, (char *)(this)));
         }
      }
      _v.resize(vectorSize, ' ');
      _l.clear();
   }
   std::list<std::pair<std::size_t, char *> > _l;
   std::vector<char> _v;
};
```

Understanding Overhead: A Simulation Utility Class



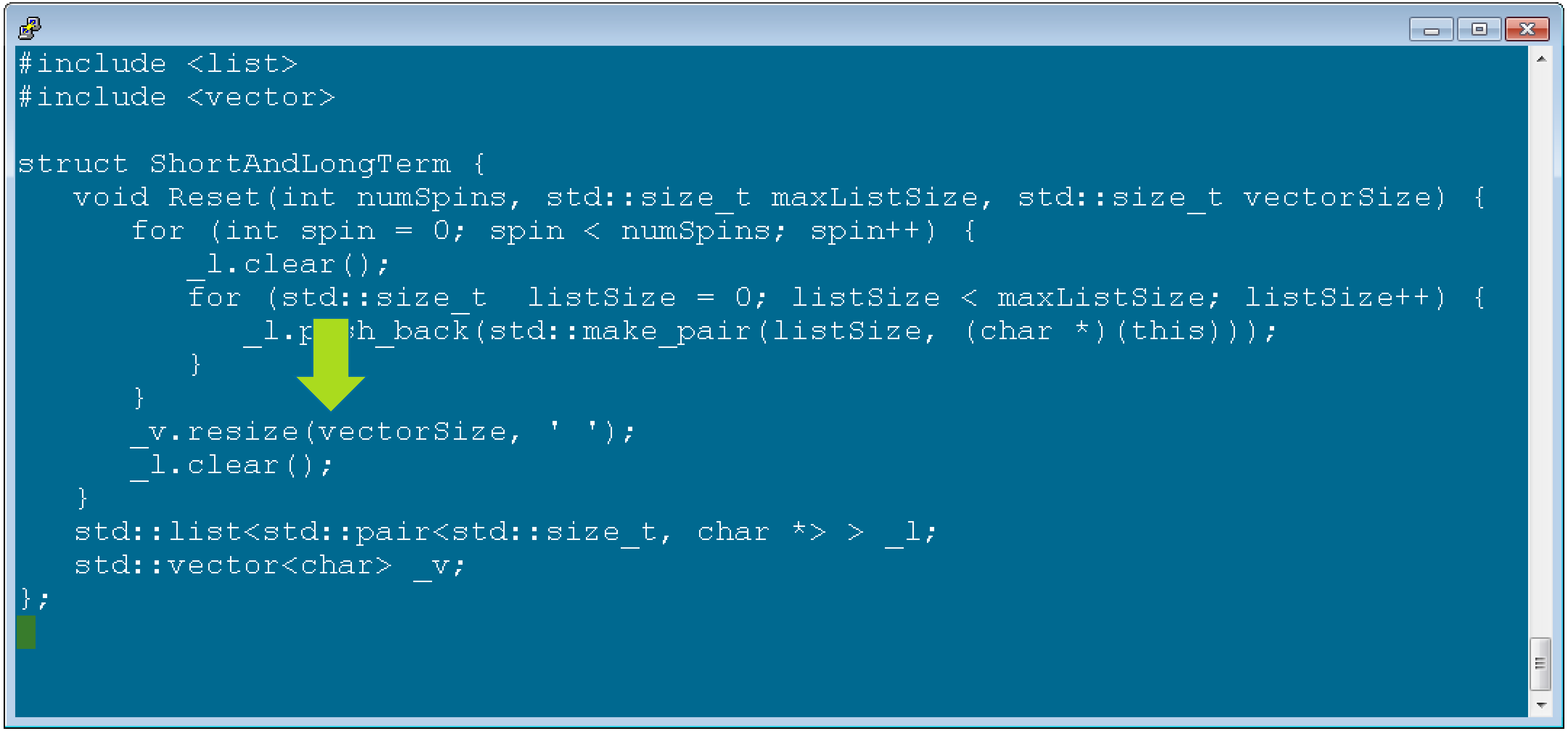
```
#include <list>
#include <vector>

struct ShortAndLongTerm {
   void Reset(int numSpins, std::size_t maxListSize, std::size_t vectorSize) {
      for (int spin = 0; spin < numSpins; spin++) {
         _l.clear();
         for (std::size_t  listSize = 0; listSize < maxListSize; listSize++) {
            _l.push_back(std::make_pair(listSize, (char *)(this)));
         }
      }
      _v.resize(vectorSize, ' ');
      _l.clear();
   }
   std::list<std::pair<std::size_t, char *> > _l;
   std::vector<char> _v;
};
```

Understanding Overhead: A Simulation Utility Class



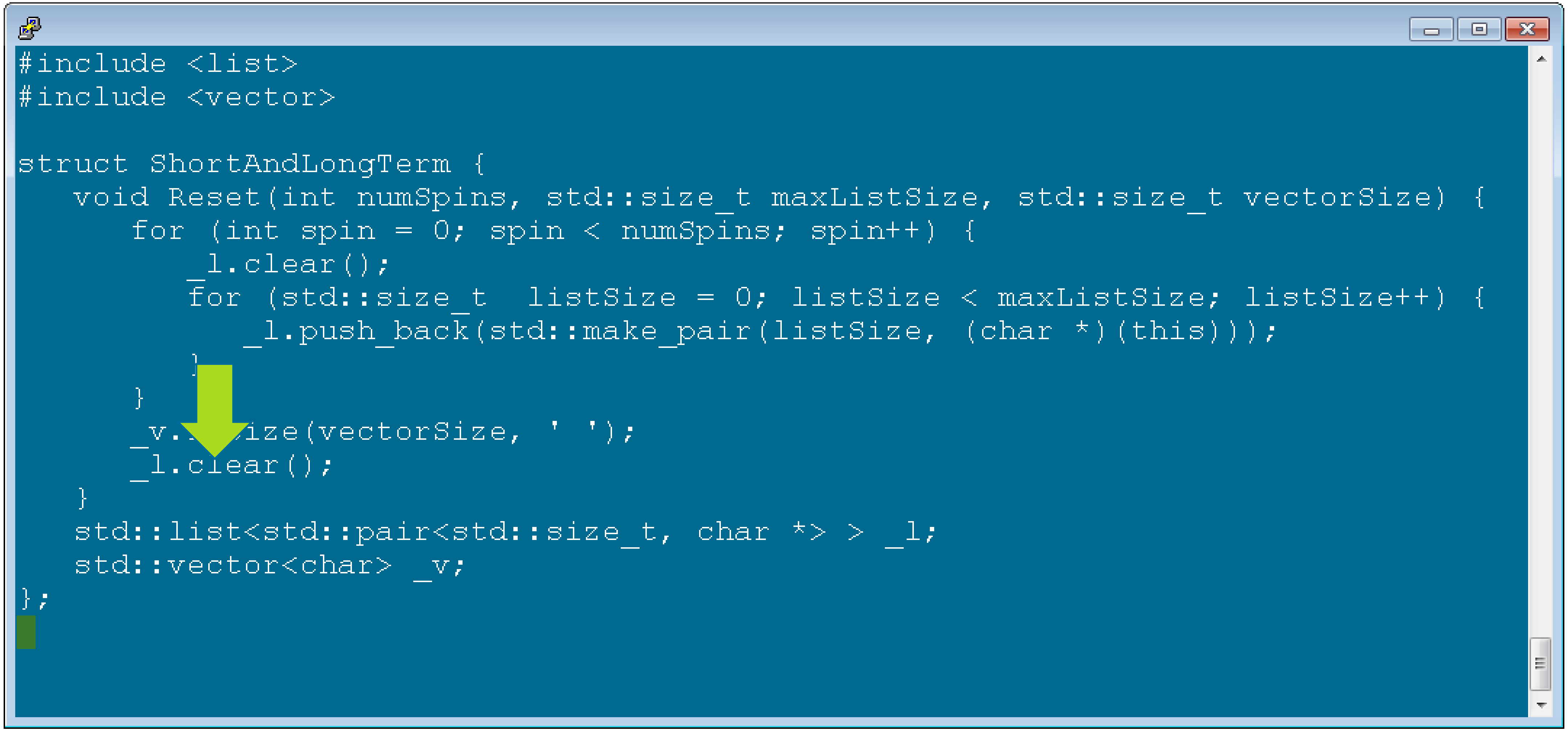
```
#include <list>
#include <vector>

struct ShortAndLongTerm {
   void Reset(int numSpins, std::size_t maxListSize, std::size_t vectorSize) {
      for (int spin = 0; spin < numSpins; spin++) {
         _l.clear();
         for (std::size_t listSize = 0; listSize < maxListSize; listSize++) {
            _l.push_back(std::make_pair(listSize, (char *)(this)));
         }
      }
      _v.resize(vectorSize, ' ');
      _l.clear();
   }
   std::list<std::pair<std::size_t, char *> > _l;
   std::vector<char> _v;
};
```

Understanding Overhead: A Simulation Utility Class



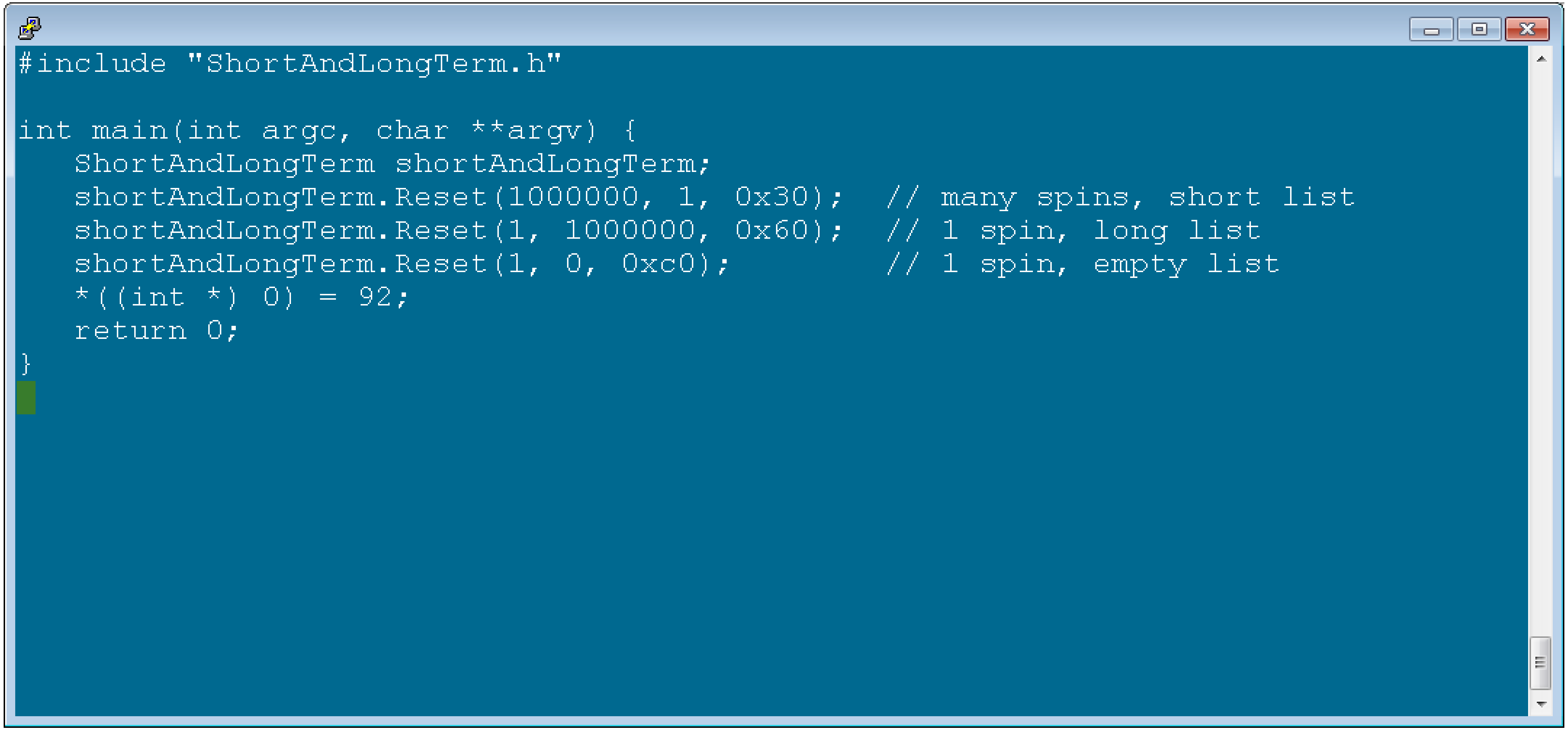
```
#include <list>
#include <vector>

struct ShortAndLongTerm {
   void Reset(int numSpins, std::size_t maxListSize, std::size_t vectorSize) {
      for (int spin = 0; spin < numSpins; spin++) {
         _l.clear();
         for (std::size_t  listSize = 0; listSize < maxListSize; listSize++) {
            _l.push_back(std::make_pair(listSize, (char *)(this)));
         }
      }
      _v.resize(vectorSize, ' ');
      _l.clear();
   }
   std::list<std::pair<std::size_t, char *> > _l;
   std::vector<char> _v;
};
```

Understanding Overhead: A Simulation Utility Class



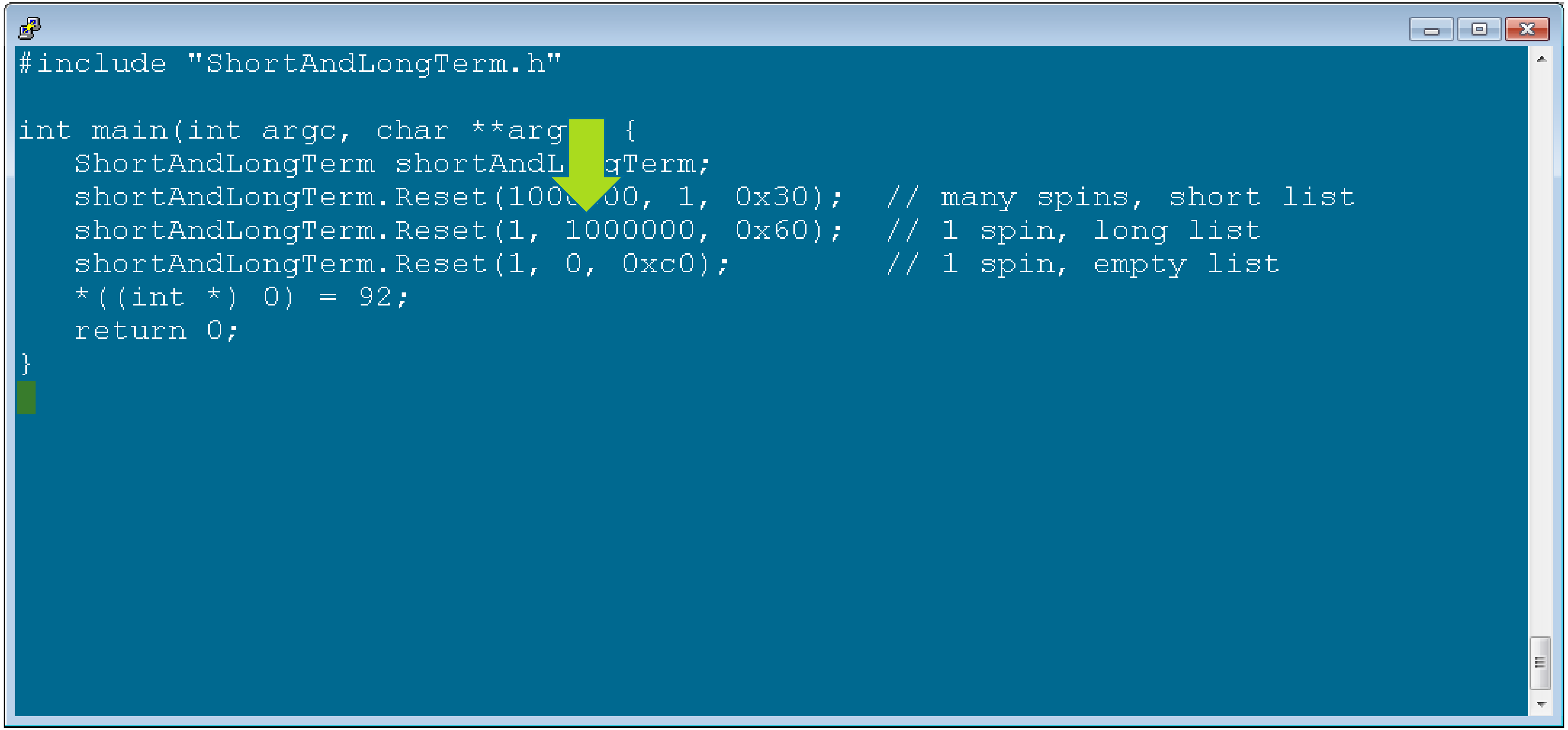
```
#include <list>
#include <vector>

struct ShortAndLongTerm {
   void Reset(int numSpins, std::size_t maxListSize, std::size_t vectorSize) {
      for (int spin = 0; spin < numSpins; spin++) {
         _l.clear();
         for (std::size_t  listSize = 0; listSize < maxListSize; listSize++) {
            _l.push_back(std::make_pair(listSize, (char *)(this)));
         }
      }
      _v.resize(vectorSize, ' ');
      _l.clear();
   }
   std::list<std::pair<std::size_t, char *> > _l;
   std::vector<char> _v;
};
```

Understanding Overhead: A Simulation Class



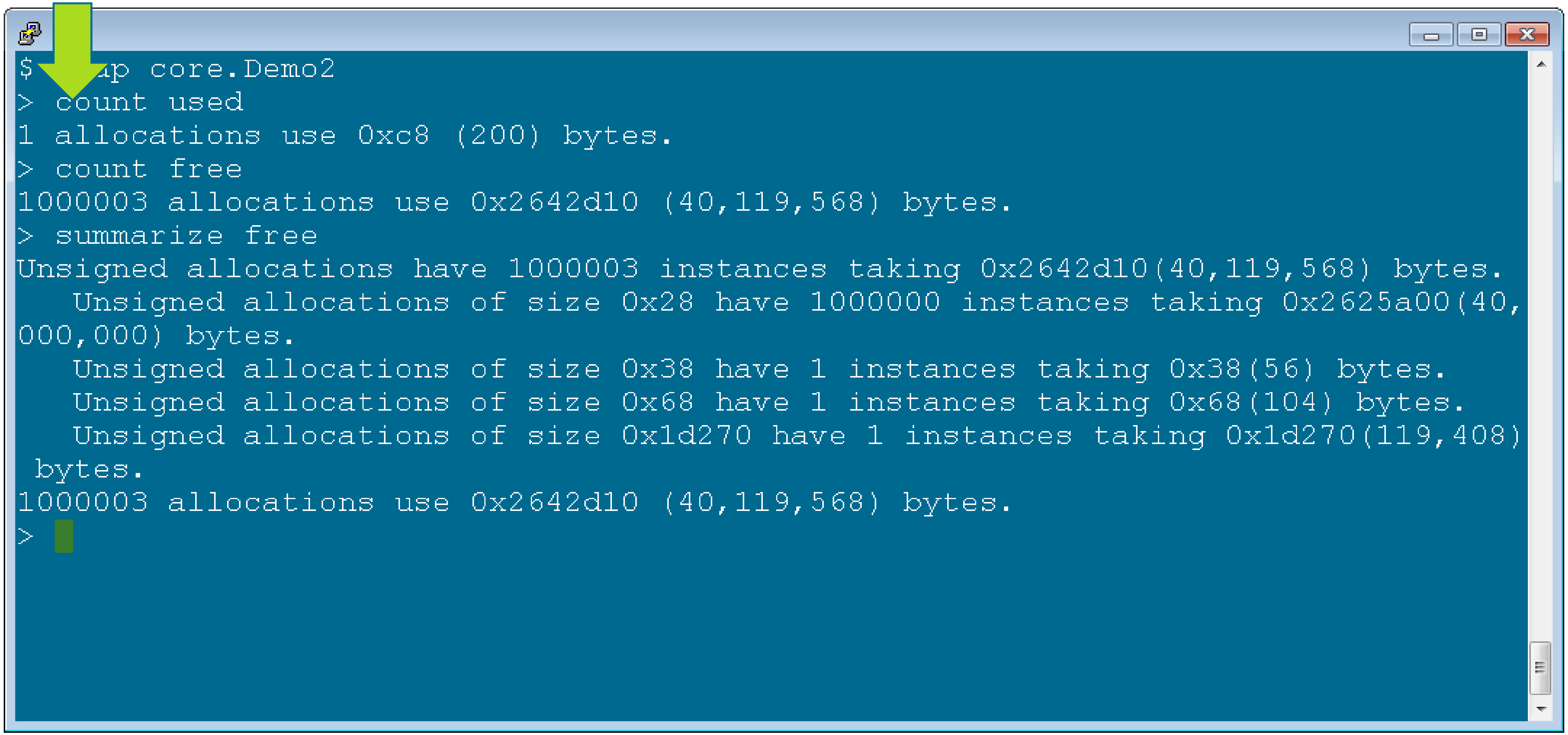
```
#include "ShortAndLongTerm.h"

int main(int argc, char **argv) {
   ShortAndLongTerm shortAndLongTerm;
   shortAndLongTerm.Reset(1000000, 1, 0x30);  // many spins, short list
   shortAndLongTerm.Reset(1, 1000000, 0x60);  // 1 spin, long list
   shortAndLongTerm.Reset(1, 0, 0xc0);        // 1 spin, empty list
   *((int *) 0) = 92;
   return 0;
}
```

Understanding Overhead: A Simulation Class



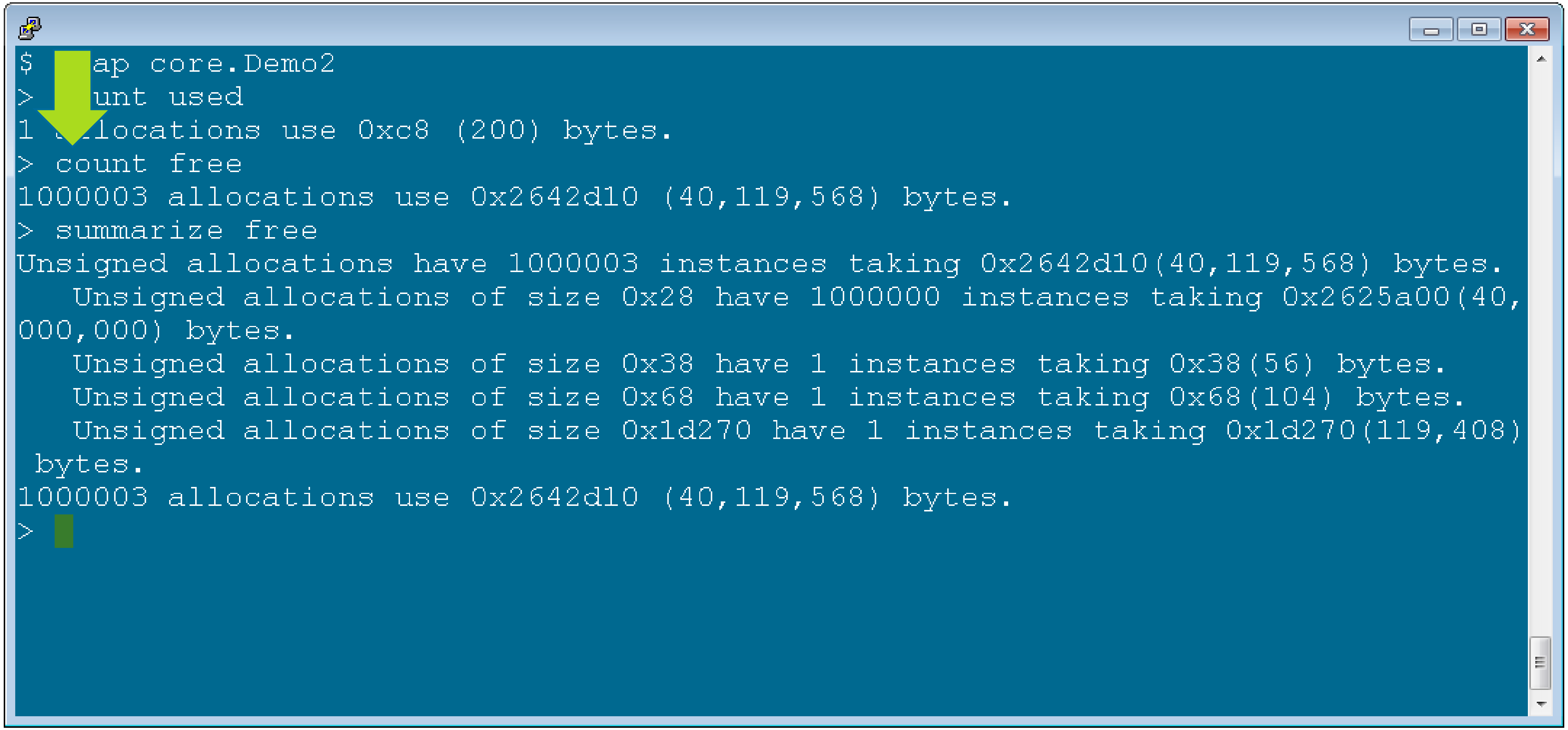
```
#include "ShortAndLongTerm.h"

int main(int argc, char **argv) {
   ShortAndLongTerm shortAndLongTerm;
   shortAndLongTerm.Reset(1000000, 1, 0x30);   // many spins, short list
   shortAndLongTerm.Reset(1, 1000000, 0x60);   // 1 spin, long list
   shortAndLongTerm.Reset(1, 0, 0xc0);         // 1 spin, empty list
   *((int *) 0) = 92;
   return 0;
}
```

Understanding Overhead: Looking at the Core



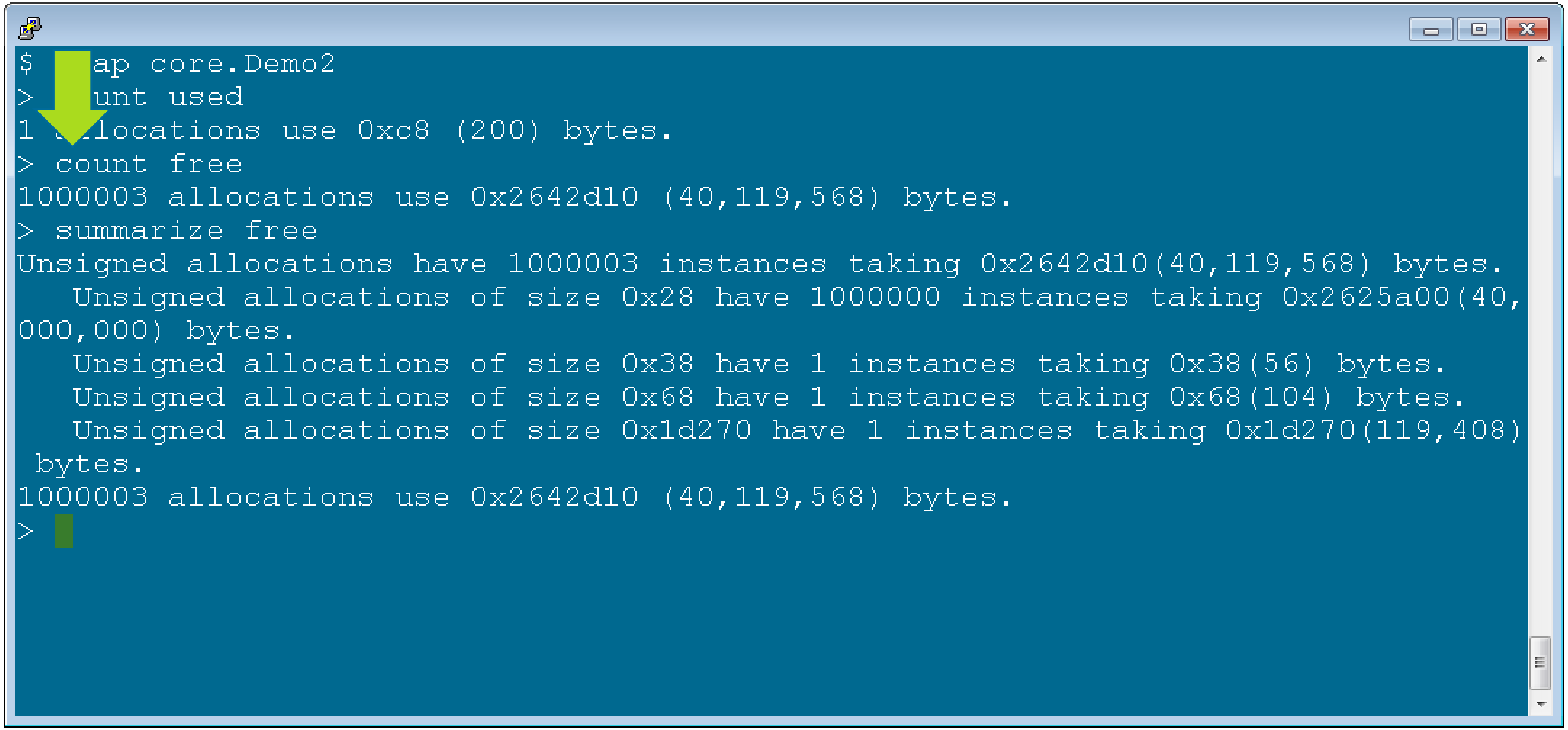
```
$ chap core.Demo2
> count used
1 allocations use 0xc8 (200) bytes.
> count free
1000003 allocations use 0x2642d10 (40,119,568) bytes.
> summarize free
Unsigned allocations have 1000003 instances taking 0x2642d10(40,119,568) bytes.
   Unsigned allocations of size 0x28 have 1000000 instances taking 0x2625a00(40,
000,000) bytes.
   Unsigned allocations of size 0x38 have 1 instances taking 0x38(56) bytes.
   Unsigned allocations of size 0x68 have 1 instances taking 0x68(104) bytes.
   Unsigned allocations of size 0x1d270 have 1 instances taking 0x1d270(119,408)
 bytes.
1000003 allocations use 0x2642d10 (40,119,568) bytes.
> 
```

Understanding Overhead: Looking at the Core



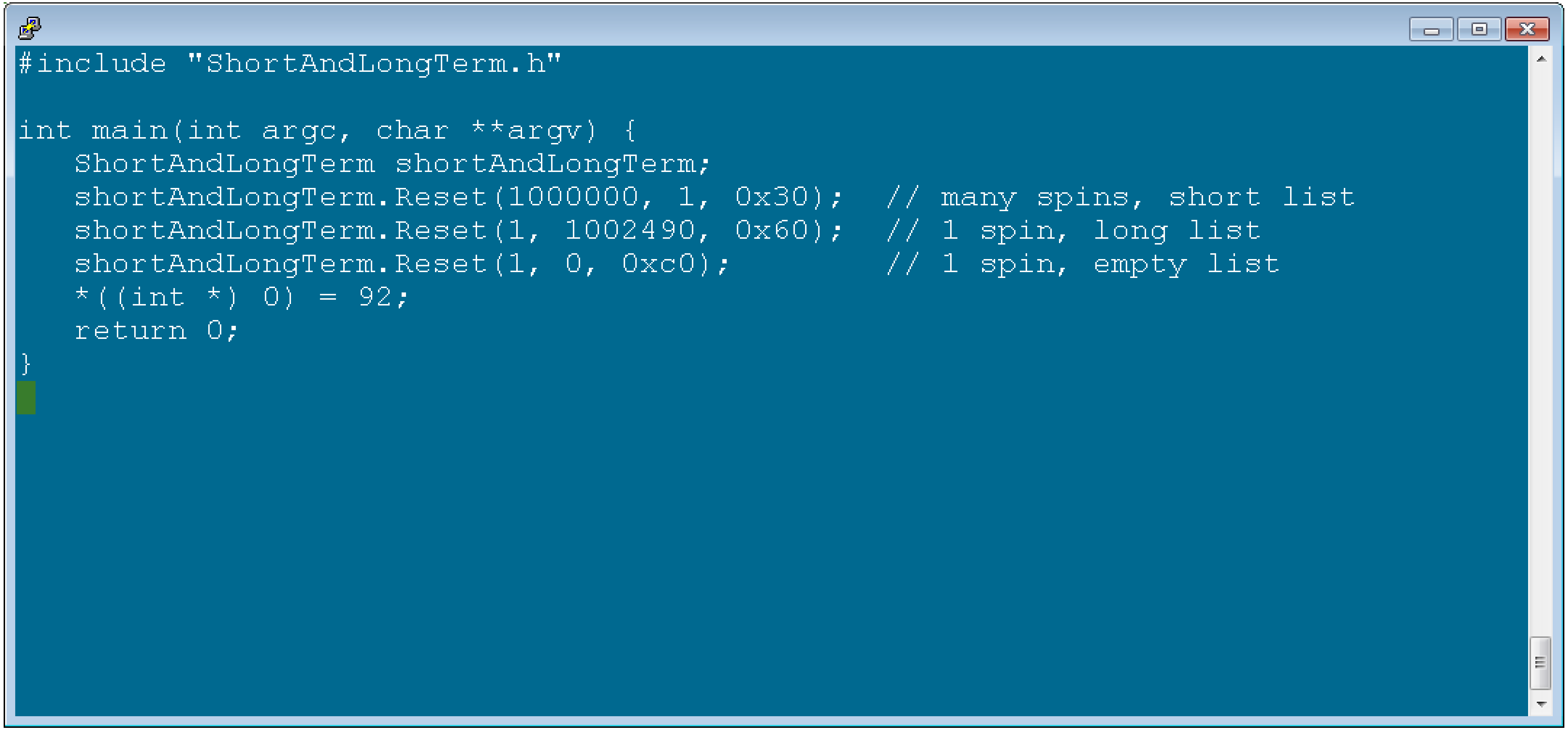
```
$   ap core.Demo2
>   unt used
1   locations use 0xc8 (200) bytes.
> count free
1000003 allocations use 0x2642d10 (40,119,568) bytes.
> summarize free
Unsigned allocations have 1000003 instances taking 0x2642d10(40,119,568) bytes.
   Unsigned allocations of size 0x28 have 1000000 instances taking 0x2625a00(40,
000,000) bytes.
   Unsigned allocations of size 0x38 have 1 instances taking 0x38(56) bytes.
   Unsigned allocations of size 0x68 have 1 instances taking 0x68(104) bytes.
   Unsigned allocations of size 0x1d270 have 1 instances taking 0x1d270(119,408)
 bytes.
1000003 allocations use 0x2642d10 (40,119,568) bytes.
> 
```

Understanding Overhead: Looking at the Core



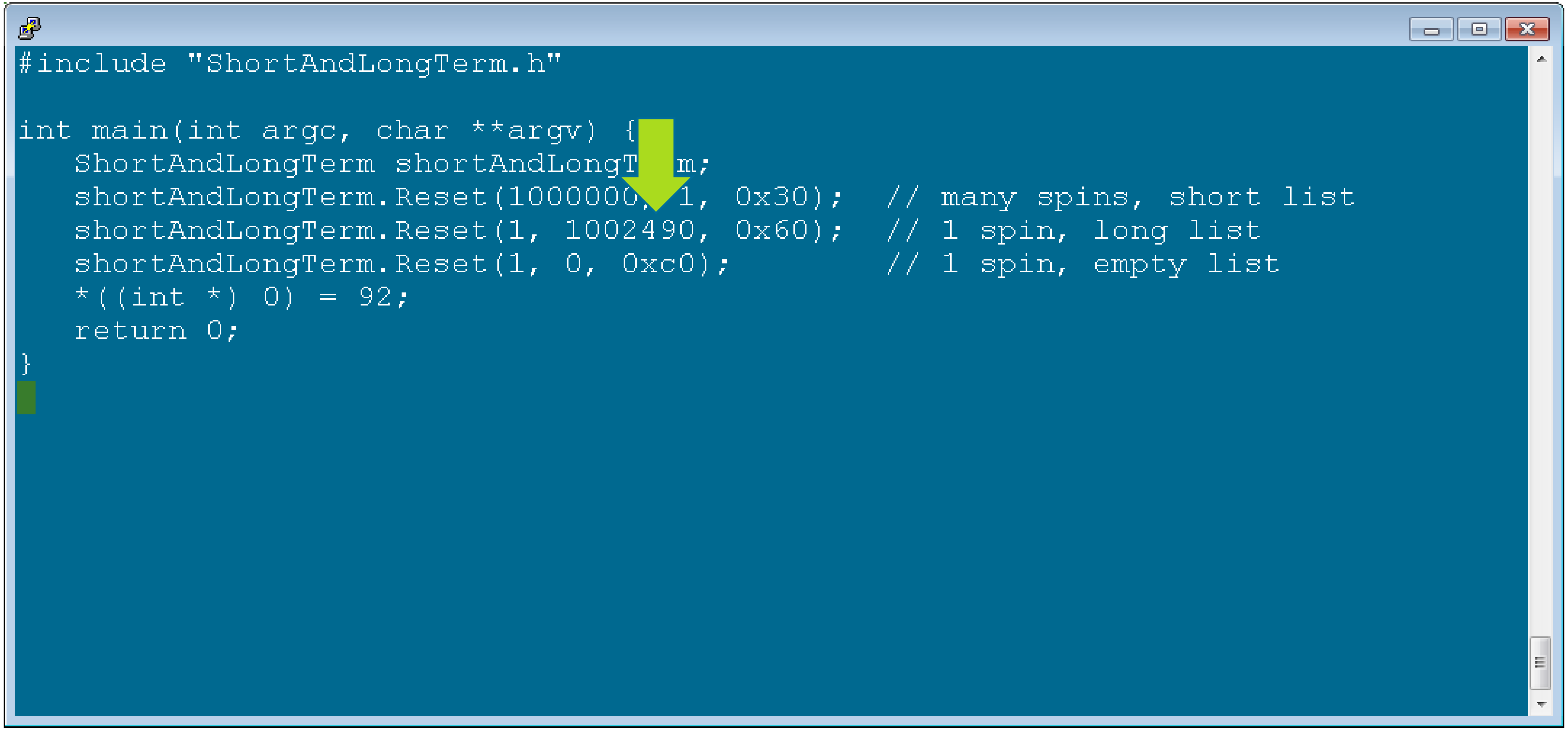
```
$    ap core.Demo2
>    unt used
1 allocations use 0xc8 (200) bytes.
> count free
1000003 allocations use 0x2642d10 (40,119,568) bytes.
> summarize free
Unsigned allocations have 1000003 instances taking 0x2642d10(40,119,568) bytes.
   Unsigned allocations of size 0x28 have 1000000 instances taking 0x2625a00(40,
000,000) bytes.
   Unsigned allocations of size 0x38 have 1 instances taking 0x38(56) bytes.
   Unsigned allocations of size 0x68 have 1 instances taking 0x68(104) bytes.
   Unsigned allocations of size 0x1d270 have 1 instances taking 0x1d270(119,408)
 bytes.
1000003 allocations use 0x2642d10 (40,119,568) bytes.
>
```

Understanding Overhead: A Similar Simulation



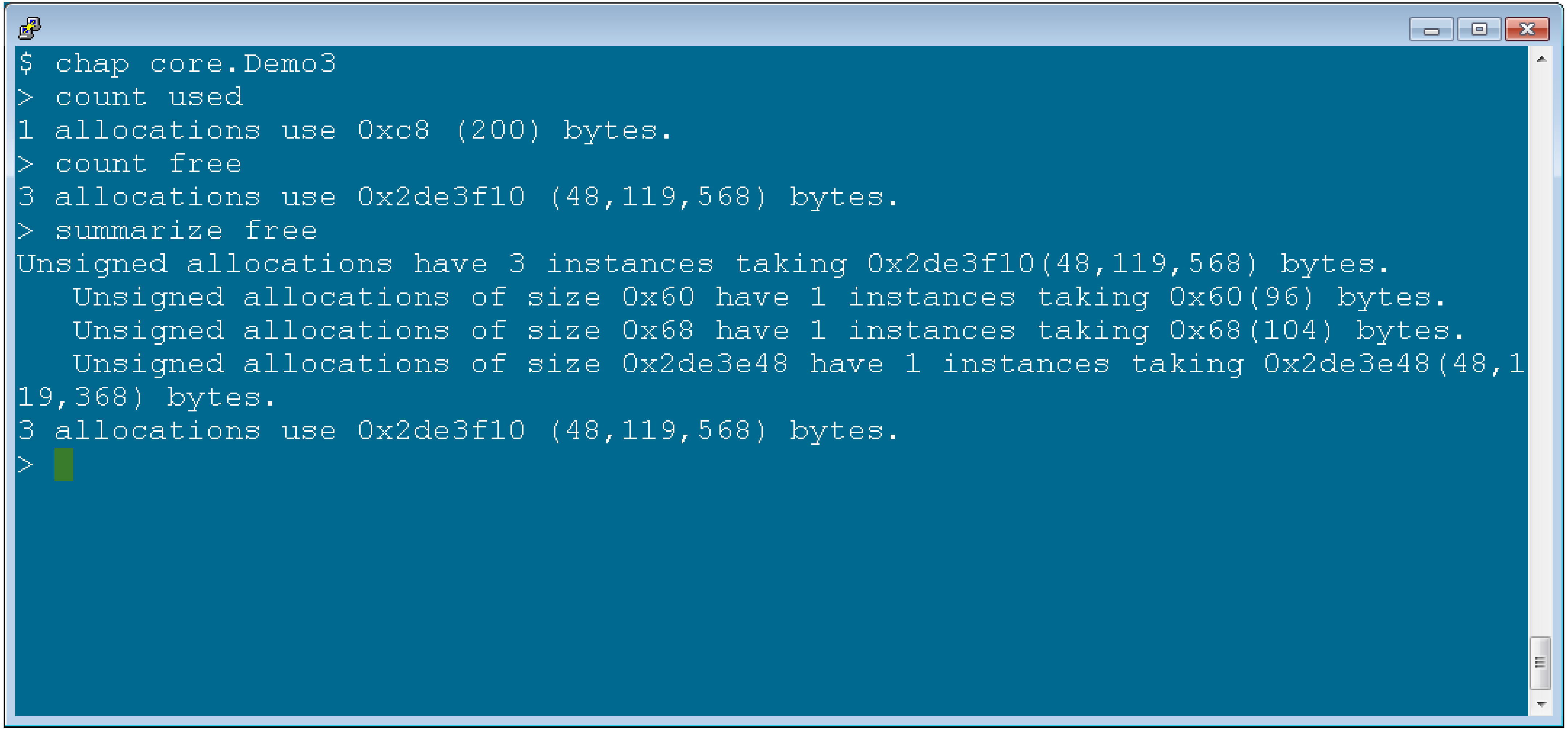
```
#include "ShortAndLongTerm.h"

int main(int argc, char **argv) {
   ShortAndLongTerm shortAndLongTerm;
   shortAndLongTerm.Reset(1000000, 1, 0x30);  // many spins, short list
   shortAndLongTerm.Reset(1, 1002490, 0x60);  // 1 spin, long list
   shortAndLongTerm.Reset(1, 0, 0xc0);        // 1 spin, empty list
   *((int *) 0) = 92;
   return 0;
}
```

Understanding Overhead: A Similar Simulation



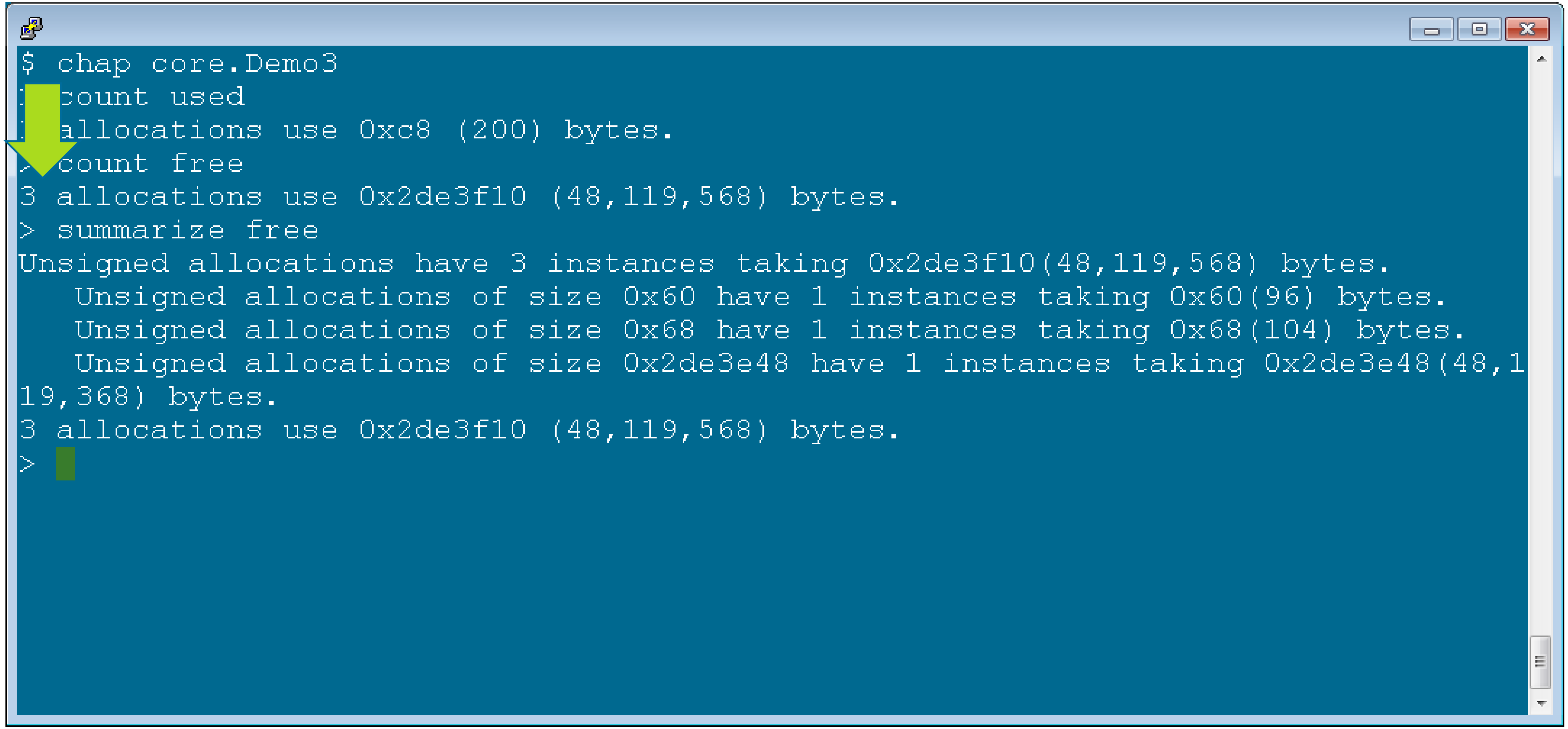
```
#include "ShortAndLongTerm.h"

int main(int argc, char **argv) {
   ShortAndLongTerm shortAndLongTerm;
   shortAndLongTerm.Reset(1000000, 1, 0x30);  // many spins, short list
   shortAndLongTerm.Reset(1, 1002490, 0x60);  // 1 spin, long list
   shortAndLongTerm.Reset(1, 0, 0xc0);        // 1 spin, empty list
   *((int *) 0) = 92;
   return 0;
}
```

Understanding Overhead: Looking at the Core



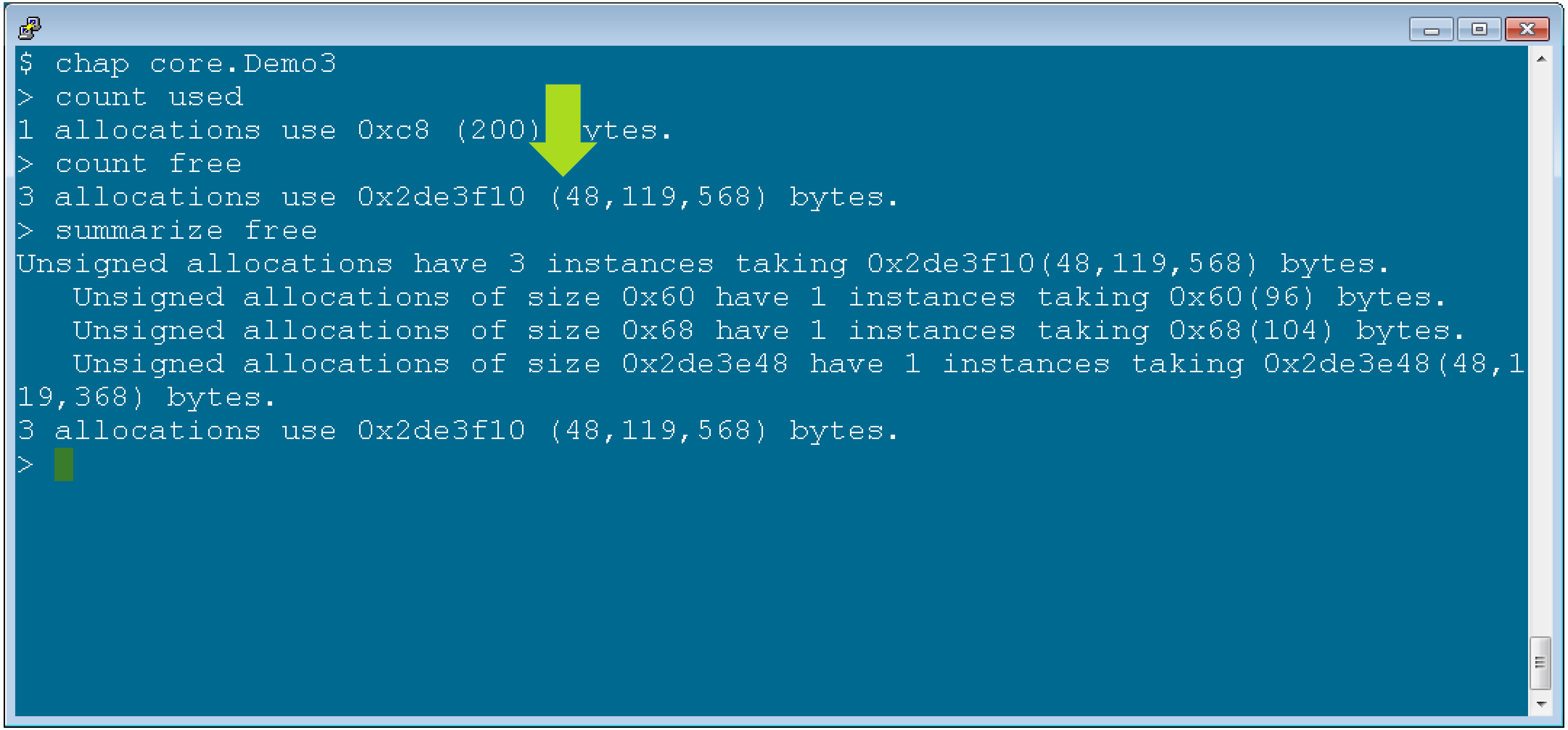
```
$ chap core.Demo3
> count used
1 allocations use 0xc8 (200) bytes.
> count free
3 allocations use 0x2de3f10 (48,119,568) bytes.
> summarize free
Unsigned allocations have 3 instances taking 0x2de3f10(48,119,568) bytes.
   Unsigned allocations of size 0x60 have 1 instances taking 0x60(96) bytes.
   Unsigned allocations of size 0x68 have 1 instances taking 0x68(104) bytes.
   Unsigned allocations of size 0x2de3e48 have 1 instances taking 0x2de3e48(48,1
19,368) bytes.
3 allocations use 0x2de3f10 (48,119,568) bytes.
>
```

Understanding Overhead: Looking at the Core



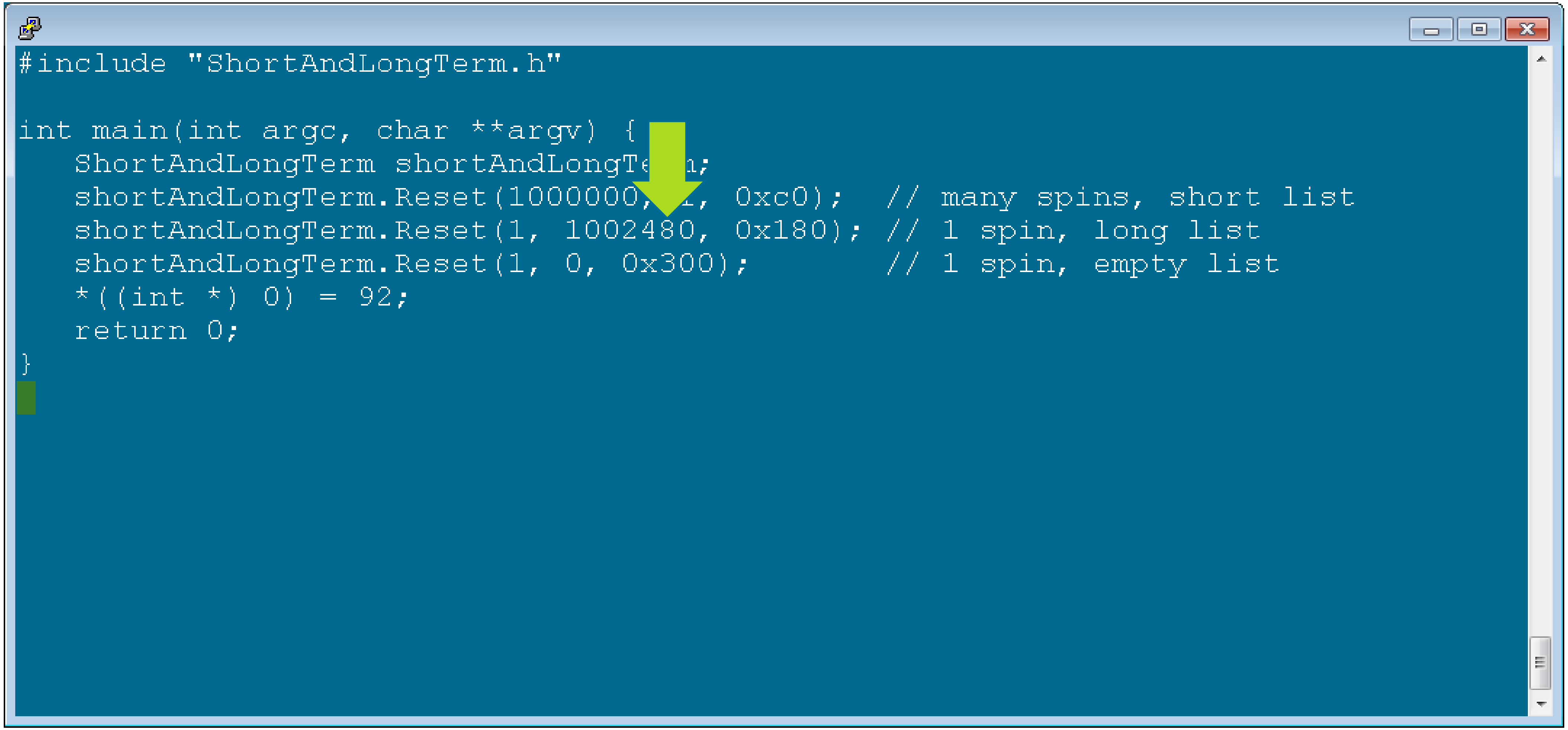
```
$ chap core.Demo3
> count used
2 allocations use 0xc8 (200) bytes.
> count free
3 allocations use 0x2de3f10 (48,119,568) bytes.
> summarize free
Unsigned allocations have 3 instances taking 0x2de3f10(48,119,568) bytes.
   Unsigned allocations of size 0x60 have 1 instances taking 0x60(96) bytes.
   Unsigned allocations of size 0x68 have 1 instances taking 0x68(104) bytes.
   Unsigned allocations of size 0x2de3e48 have 1 instances taking 0x2de3e48(48,1
19,368) bytes.
3 allocations use 0x2de3f10 (48,119,568) bytes.
>
```

Understanding Overhead: Looking at the Core



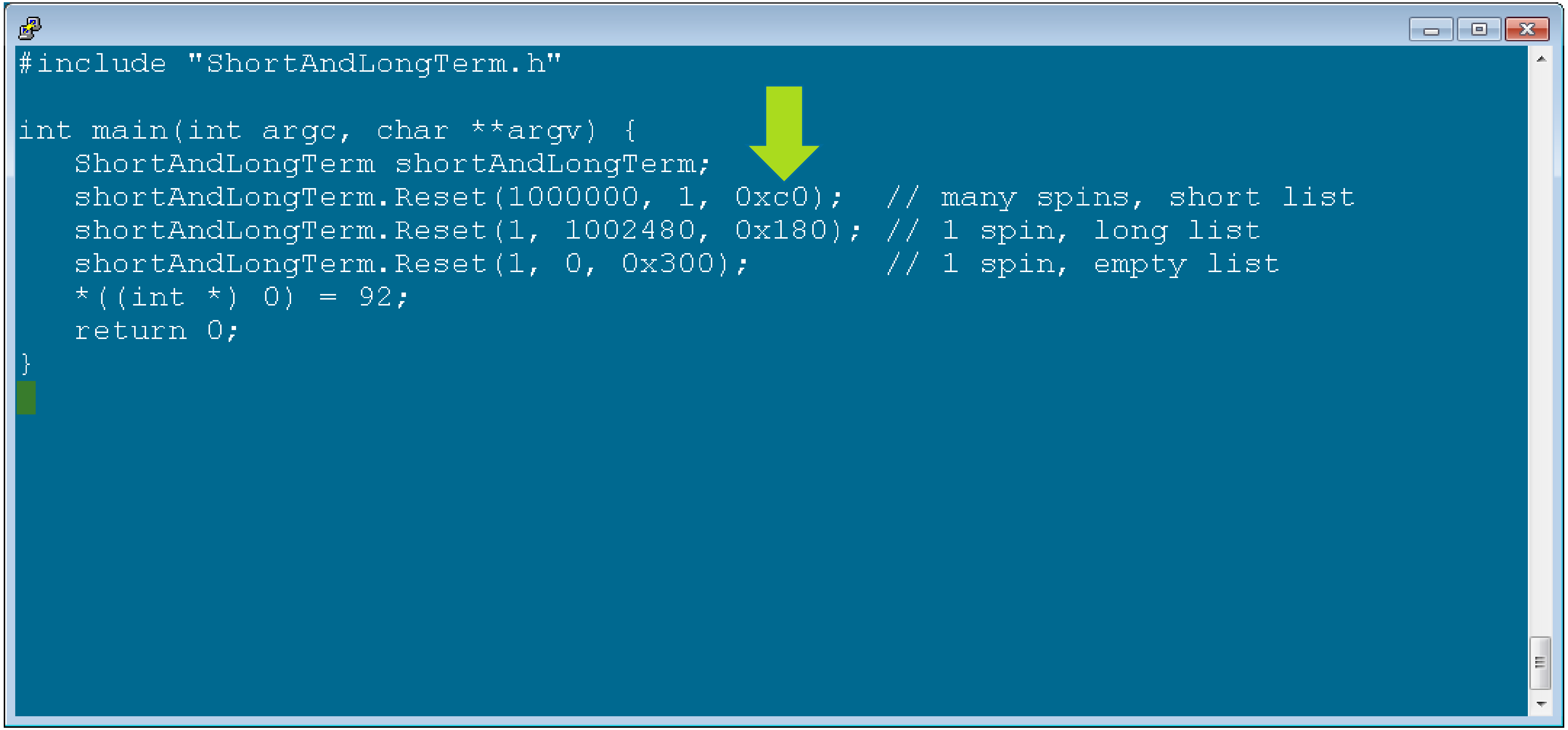
```
$ chap core.Demo3
> count used
1 allocations use 0xc8 (200)  ytes.
> count free
3 allocations use 0x2de3f10 (48,119,568) bytes.
> summarize free
Unsigned allocations have 3 instances taking 0x2de3f10(48,119,568) bytes.
   Unsigned allocations of size 0x60 have 1 instances taking 0x60(96) bytes.
   Unsigned allocations of size 0x68 have 1 instances taking 0x68(104) bytes.
   Unsigned allocations of size 0x2de3e48 have 1 instances taking 0x2de3e48(48,1
19,368) bytes.
3 allocations use 0x2de3f10 (48,119,568) bytes.
>
```

Understanding Overhead: Another Similar Simulation



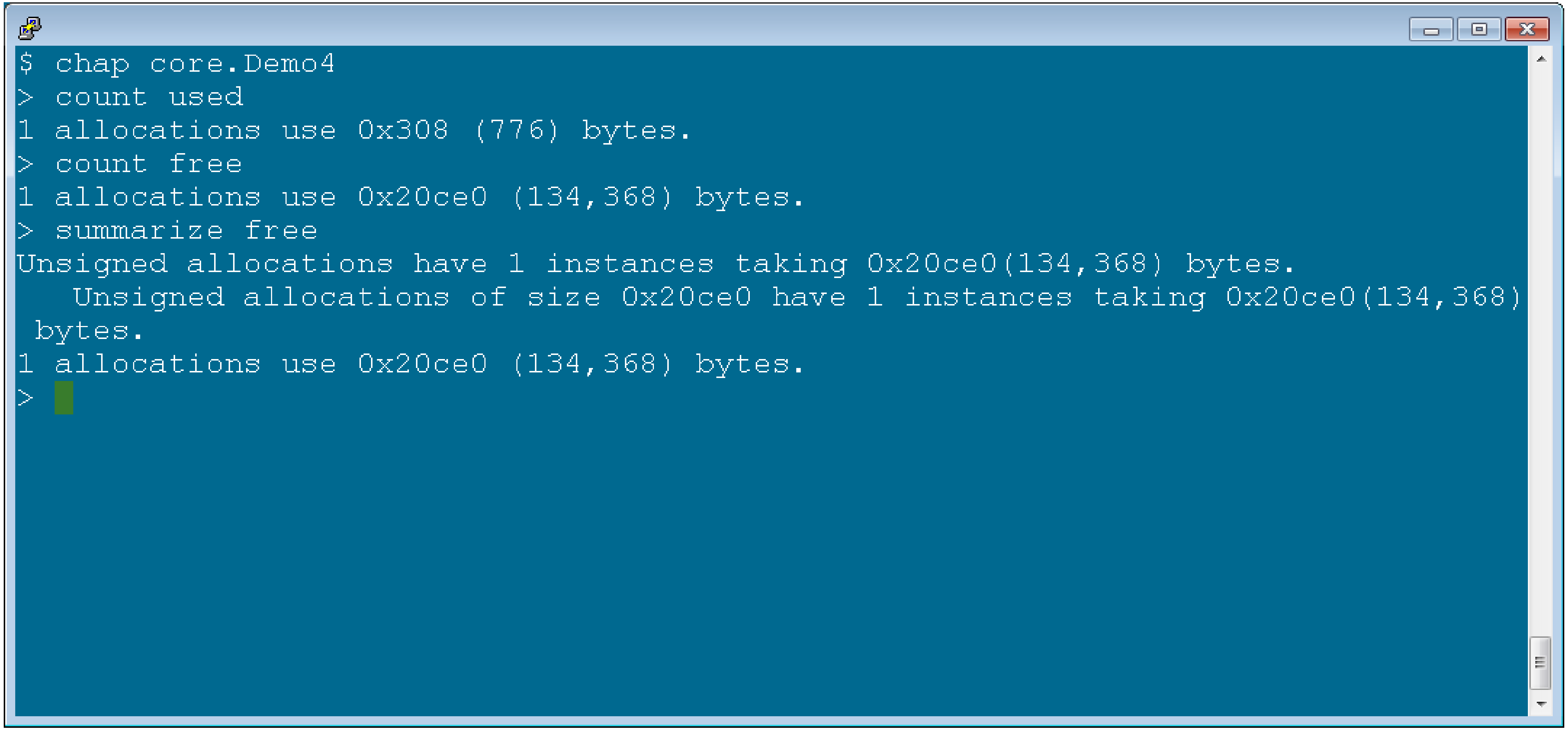
```
#include "ShortAndLongTerm.h"

int main(int argc, char **argv) {
   ShortAndLongTerm shortAndLongTerm;
   shortAndLongTerm.Reset(1000000, 1, 0xc0);  // many spins, short list
   shortAndLongTerm.Reset(1, 1002480, 0x180); // 1 spin, long list
   shortAndLongTerm.Reset(1, 0, 0x300);       // 1 spin, empty list
   *((int *) 0) = 92;
   return 0;
}
```

Understanding Overhead: Another Similar Simulation



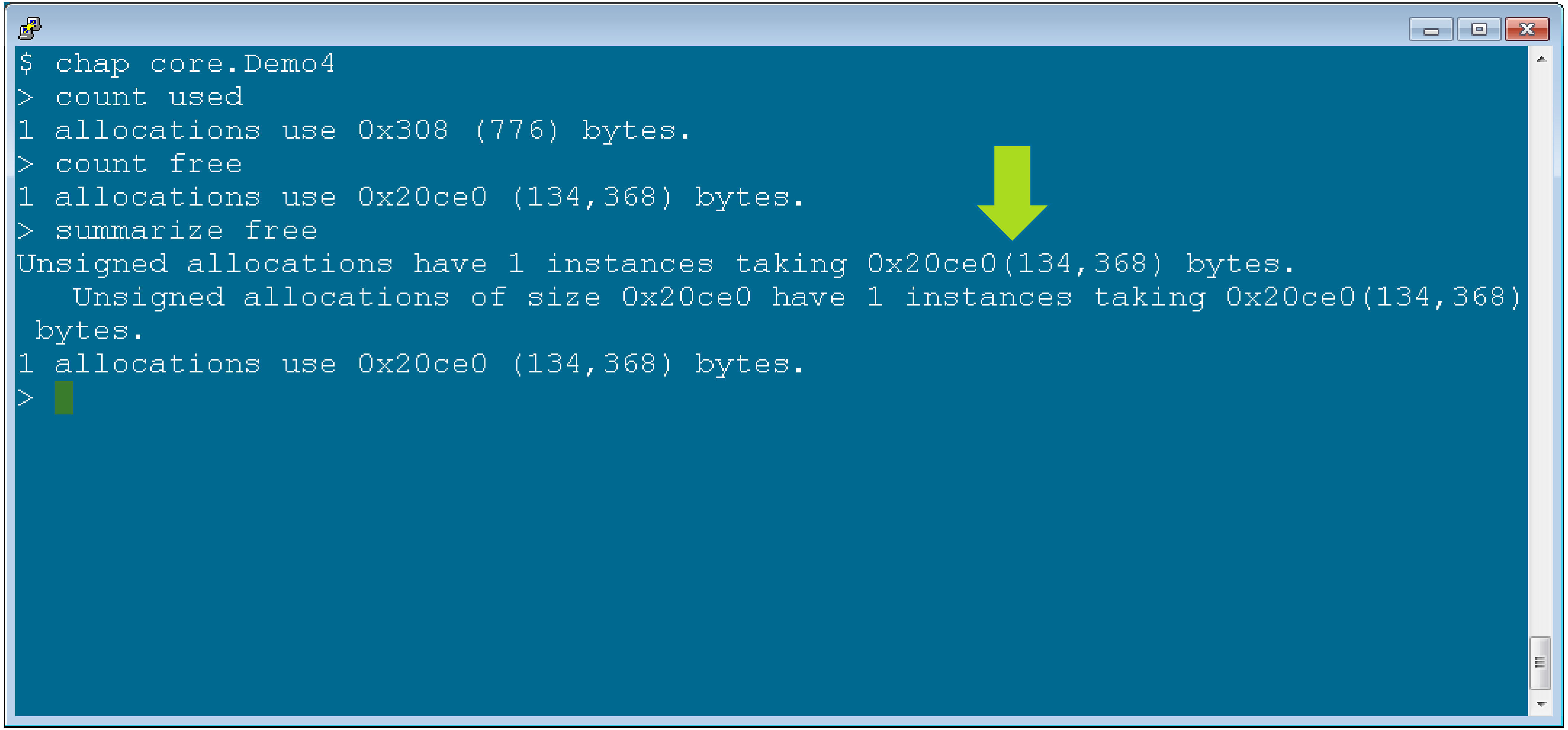
```
#include "ShortAndLongTerm.h"

int main(int argc, char **argv) {
   ShortAndLongTerm shortAndLongTerm;
   shortAndLongTerm.Reset(1000000, 1, 0xc0);  // many spins, short list
   shortAndLongTerm.Reset(1, 1002480, 0x180); // 1 spin, long list
   shortAndLongTerm.Reset(1, 0, 0x300);       // 1 spin, empty list
   *((int *) 0) = 92;
   return 0;
}
```

Understanding Overhead: Looking at the Core



```
$ chap core.Demo4
> count used
1 allocations use 0x308 (776) bytes.
> count free
1 allocations use 0x20ce0 (134,368) bytes.
> summarize free
Unsigned allocations have 1 instances taking 0x20ce0(134,368) bytes.
   Unsigned allocations of size 0x20ce0 have 1 instances taking 0x20ce0(134,368)
 bytes.
1 allocations use 0x20ce0 (134,368) bytes.
> 
```

Understanding Overhead: Looking at the Core



```
$ chap core.Demo4
> count used
1 allocations use 0x308 (776) bytes.
> count free
1 allocations use 0x20ce0 (134,368) bytes.
> summarize free
Unsigned allocations have 1 instances taking 0x20ce0(134,368) bytes.
   Unsigned allocations of size 0x20ce0 have 1 instances taking 0x20ce0(134,368)
 bytes.
1 allocations use 0x20ce0 (134,368) bytes.
> 
```

Future Directions, Q&A

- Add DWARF awareness to improve type identification and reduce false edges
- Support other allocators
  - Allocators used in production
  - Allocators used for debugging
  - Custom allocators
- Add more corruption analysis and make it more accurate
- Improve recovery in case of corruption or incomplete process images
- Add new verbs (e.g. annotate)
- Add new objects (e.g. fast bin list, allocator-specific objects)
- Add more code to identify common types and data structures

Thank You

tim@vmware.com

CHAP