Measuring Memory using valgrind

CSCE 221H

Parasol Lab, Texas A&M University

Parasol
Smarter computing.
Texas A&M University

http://parasol.tamu.edu

valgrind

- Instrumentation framework for dynamic analysis of programs (http://valgrind.org/)

`memcheck` – detecting memory errors

`cachegrind` – profiling cache / branch prediction

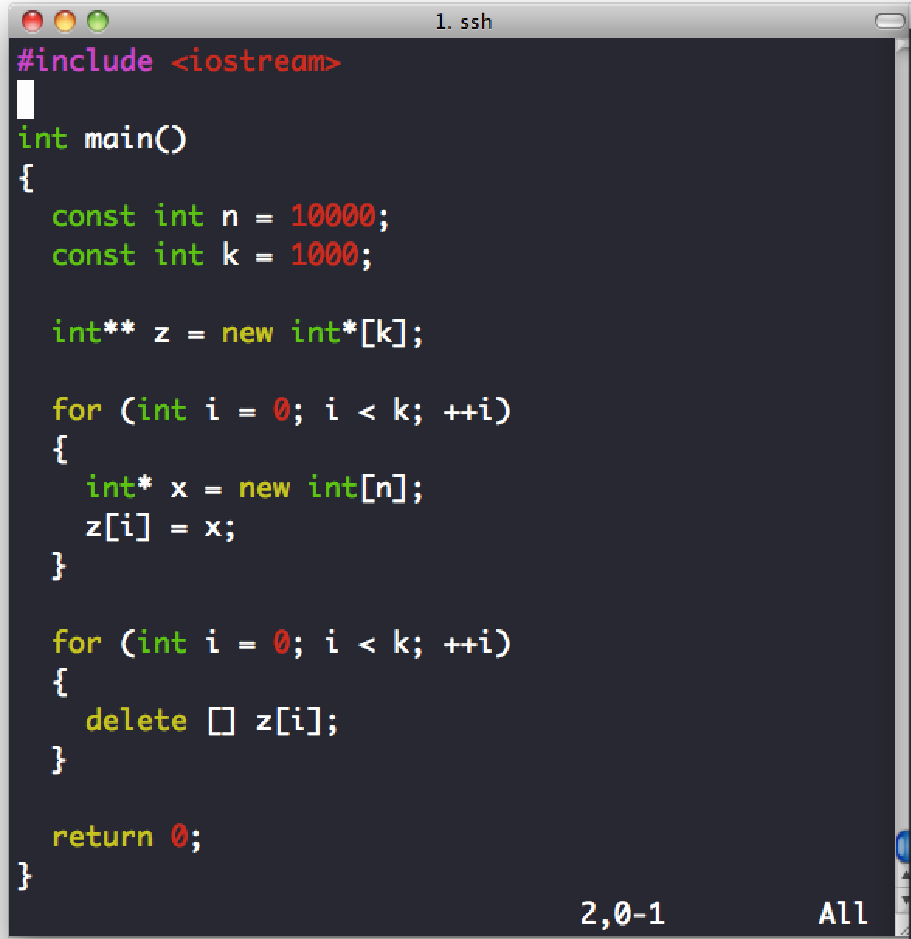
`helgrind` – detecting races in parallel programs

`massif` – profile memory allocation in heap / stack

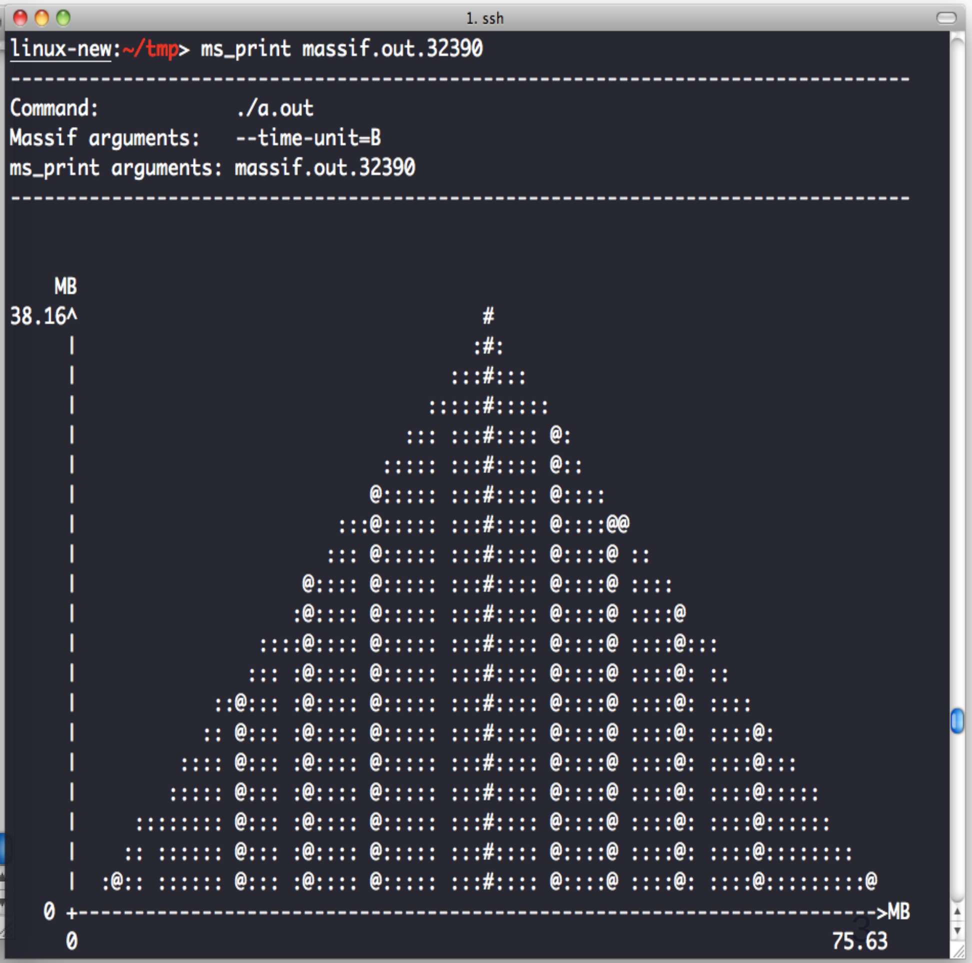
massif - heap

```
valgrind --tool=massif --time-unit=B
                ./a.out
```

```
ms_print massif.out.*
```



```
#include <iostream>

int main()
{
  const int n = 10000;
  const int k = 1000;

  int** z = new int*[k];

  for (int i = 0; i < k; ++i)
  {
    int* x = new int[n];
    z[i] = x;
  }

  for (int i = 0; i < k; ++i)
  {
    delete [] z[i];
  }

  return 0;
}
                                     2,0-1         All
```

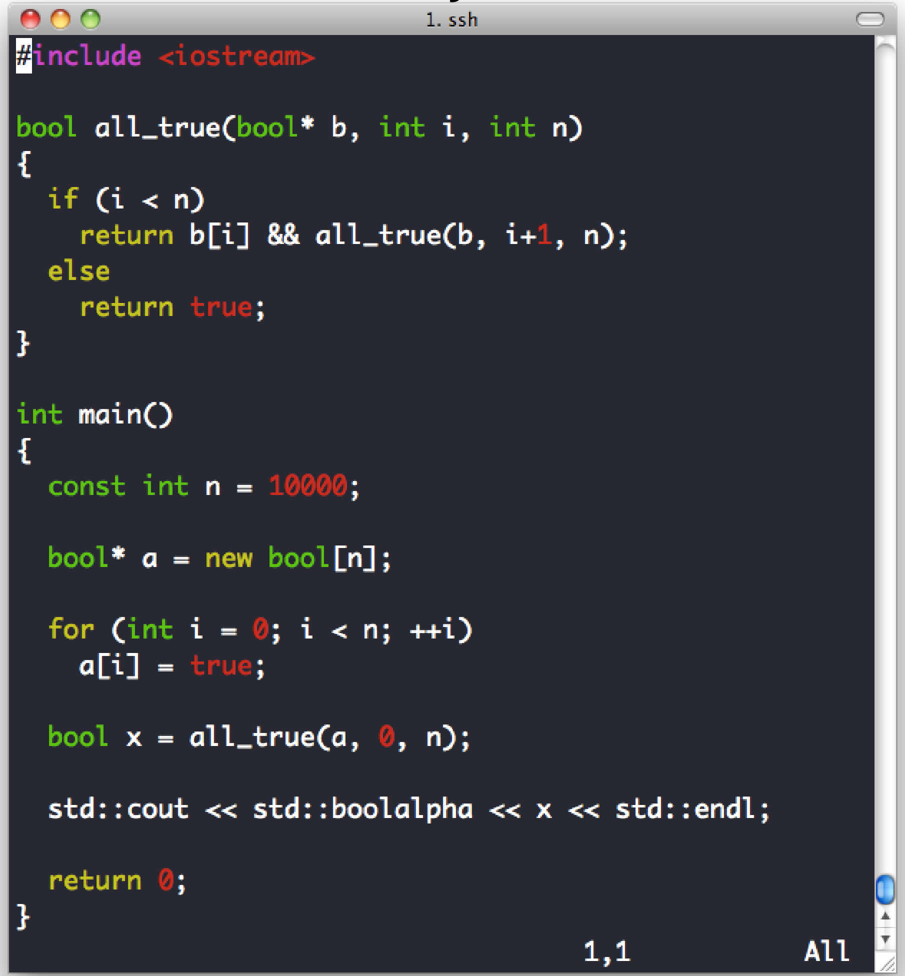


```
linux-new:~/tmp> ms_print massif.out.32390
--------------------------------------------------------------------------------
Command:            ./a.out
Massif arguments:   --time-unit=B
ms_print arguments: massif.out.32390
--------------------------------------------------------------------------------


    MB
38.16^                                    #
     |                                   :#:
     |                                 :::#:::
     |                               :::::#:::::
     |                             ::: :::#:::: @:
     |                           ::::: :::#:::: @::
     |                          @::::: :::#:::: @::::
     |                       :::@::::: :::#:::: @::::@@
     |                      ::: @::::: :::#:::: @::::@ ::
     |                    @:::: @::::: :::#:::: @::::@ ::::
     |                   :@:::: @::::: :::#:::: @::::@ ::::@
     |                ::::@:::: @::::: :::#:::: @::::@ ::::@:::
     |               ::: :@:::: @::::: :::#:::: @::::@ ::::@: ::
     |            ::@::: :@:::: @::::: :::#:::: @::::@ ::::@: ::::
     |           :: @::: :@:::: @::::: :::#:::: @::::@ ::::@: ::::@:
     |         :::: @::: :@:::: @::::: :::#:::: @::::@ ::::@: ::::@:::
     |        ::::: @::: :@:::: @::::: :::#:::: @::::@ ::::@: ::::@:::::
     |     :::::::: @::: :@:::: @::::: :::#:::: @::::@ ::::@: ::::@::::::
     |    :: :::::: @::: :@:::: @::::: :::#:::: @::::@ ::::@: ::::@::::::::
     |  :@:: :::::: @::: :@:::: @::::: :::#:::: @::::@ ::::@: ::::@:::::::::@
   0 +----------------------------------------------------------------------->MB
     0                                                                  75.63
```

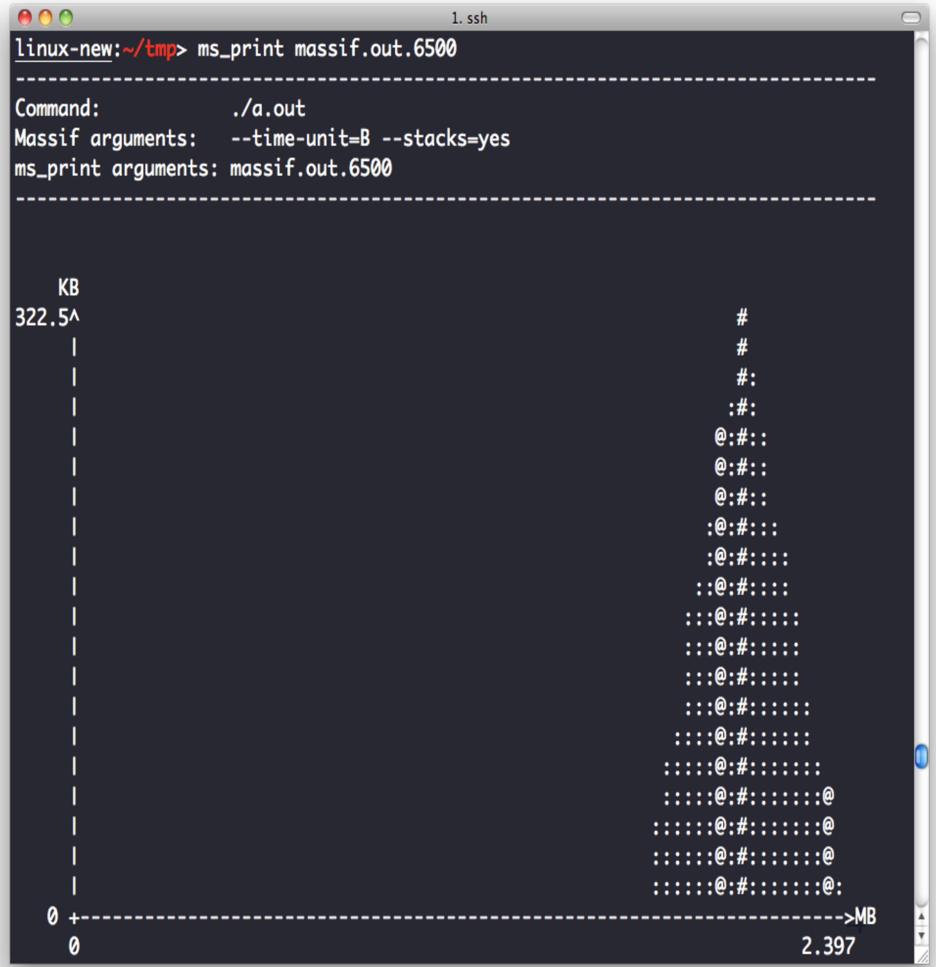
massif - stack

```
valgrind --tool=massif --time-unit=B
         --stacks=yes ./a.out
```

```
ms_print massif.out.*
```



```
#include <iostream>

bool all_true(bool* b, int i, int n)
{
  if (i < n)
    return b[i] && all_true(b, i+1, n);
  else
    return true;
}

int main()
{
  const int n = 10000;

  bool* a = new bool[n];

  for (int i = 0; i < n; ++i)
    a[i] = true;

  bool x = all_true(a, 0, n);

  std::cout << std::boolalpha << x << std::endl;

  return 0;
}
                                     1,1          All
```



```
linux-new:~/tmp> ms_print massif.out.6500
--------------------------------------------------------------------------------
Command:            ./a.out
Massif arguments:   --time-unit=B --stacks=yes
ms_print arguments: massif.out.6500
--------------------------------------------------------------------------------


    KB
322.5^                                                              #
     |                                                              #
     |                                                              #:
     |                                                             :#:
     |                                                            @:#::
     |                                                            @:#::
     |                                                            @:#::
     |                                                           :@:#:::
     |                                                           :@:#::::
     |                                                          ::@:#::::
     |                                                         :::@:#:::::
     |                                                         :::@:#:::::
     |                                                         :::@:#:::::
     |                                                         :::@:#::::::
     |                                                        ::::@:#::::::
     |                                                       :::::@:#:::::::
     |                                                       :::::@:#:::::::@
     |                                                      ::::::@:#:::::::@
     |                                                      ::::::@:#:::::::@
     |                                                      ::::::@:#:::::::@:
   0 +----------------------------------------------------------------------->MB
     0                                                                   2.397
```

Exercise

1. Write an iterative version of the all_true algorithm
   - Measure memory consumption using massif of both the iterative and recursive versions
2. Plot peak memory consumption for the following:
   - n = 10^1, 10^2, 10^3, 10^4, 10^5