**LAB 2, 1/17/2019 MCS 253P**

**Name: Xiaowei Tan**

**Partner: Ethel Hoshi**

**General Problem Description**

Write a program to parseCmd shell commands into parts. Handle commands and arguments (words) and the operators <>&|.

**Additional Problem Specifics**

Q: Confused about the input, how to read arguments when they are passed by pipe

A: The input passed by pipe will be read into the program by standard input.

**Sample Input**

cat rolodex. c| tr A-Z a-z>output.foo&

**Proposed Algorithm**

***Description:***

Every time it takes current character and next character into consideration.

Firstly, if the character is a space, it does nothing and for others, it output the character.

Secondly, it decides whether to output an line feed. In the following two situations it will do nothing:

1. Current character is space.
2. Current character is not space and operators. Either for next character.

In other cases, it outputs a line feed.

***Correctness:***

It is simply a one-pass solution. In each step, it considers the current character and next character. It is a simple finite state machine, choosing what to next by the two characters.

***Time Complexity:***

O(n)

***Space Complexity:***

O(1)

**C++ Implementation of Algorithm**

for(int i = 0; i < line.length(); i++){

switch(line[i]){

case ' ':

break;

default:

cout << line[i];break;

}

linefeed(line, i);

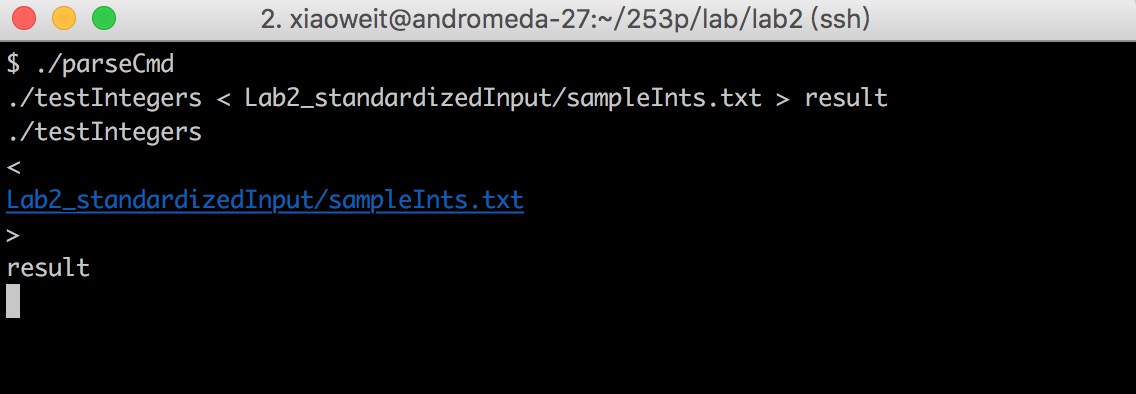
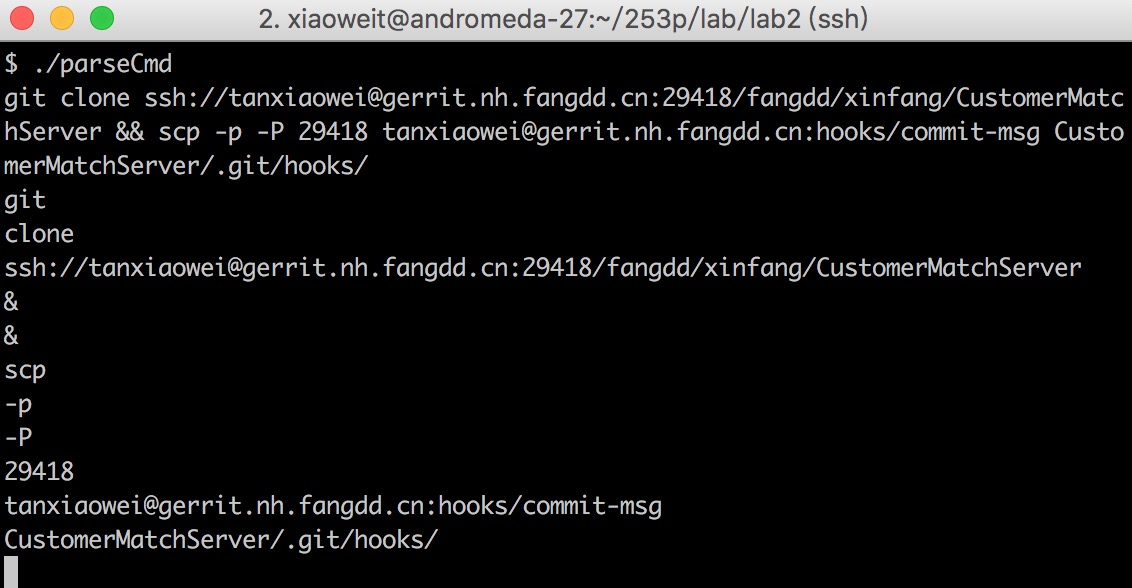
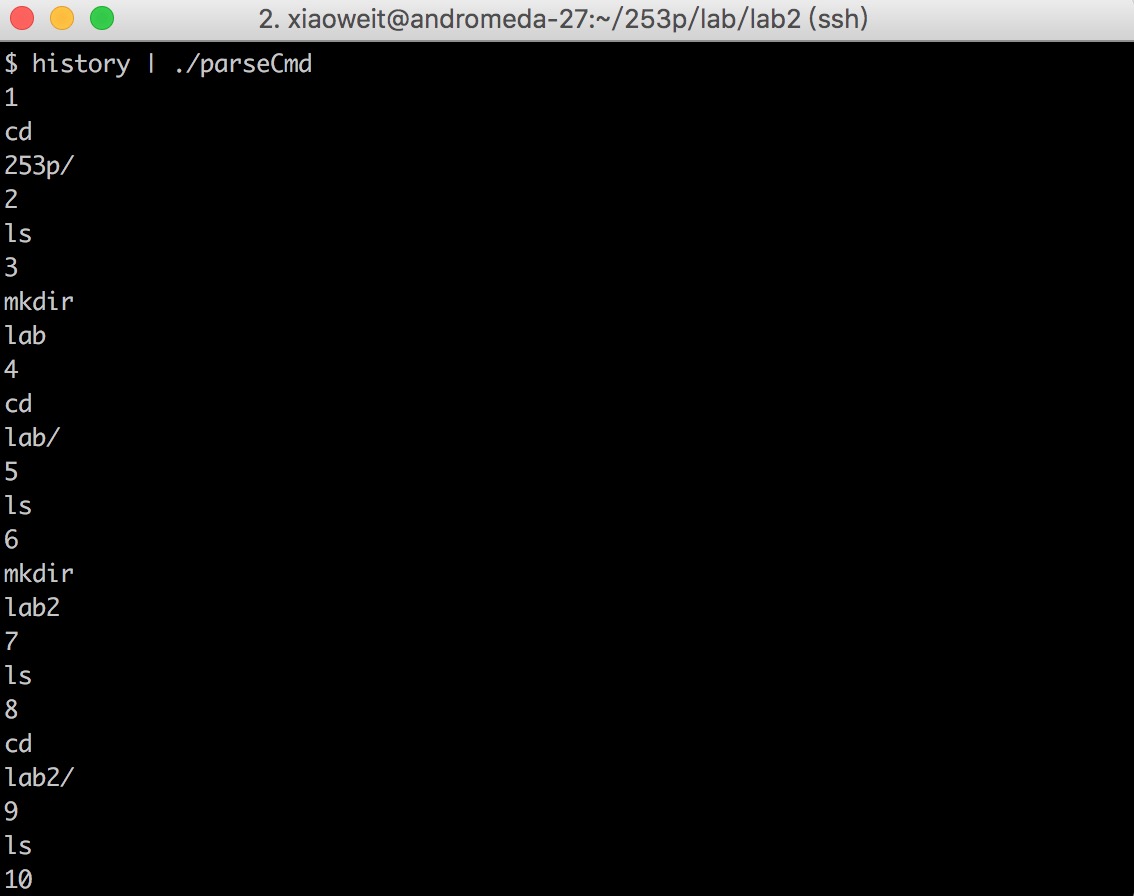
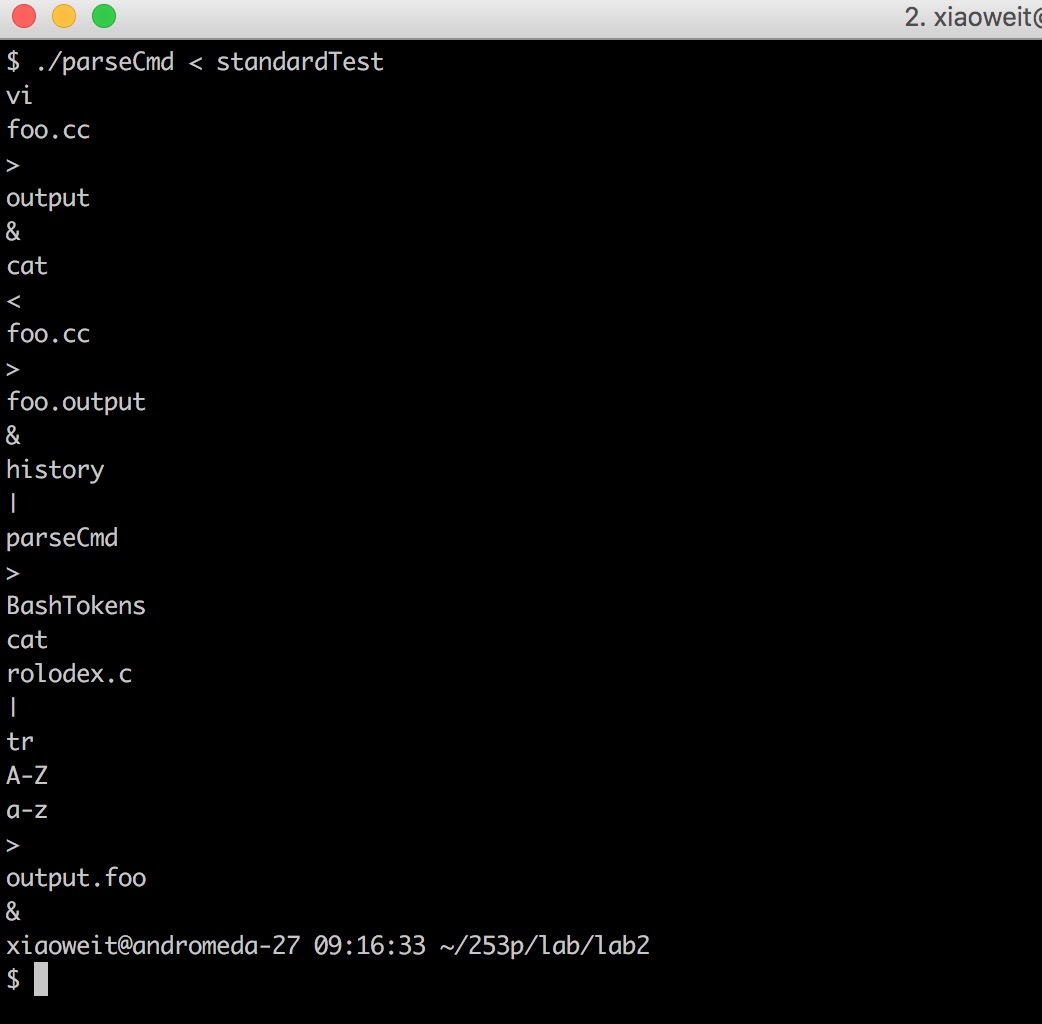
}

**Advantages/Disadvantages of Your Algorithm and Any Other Comments**

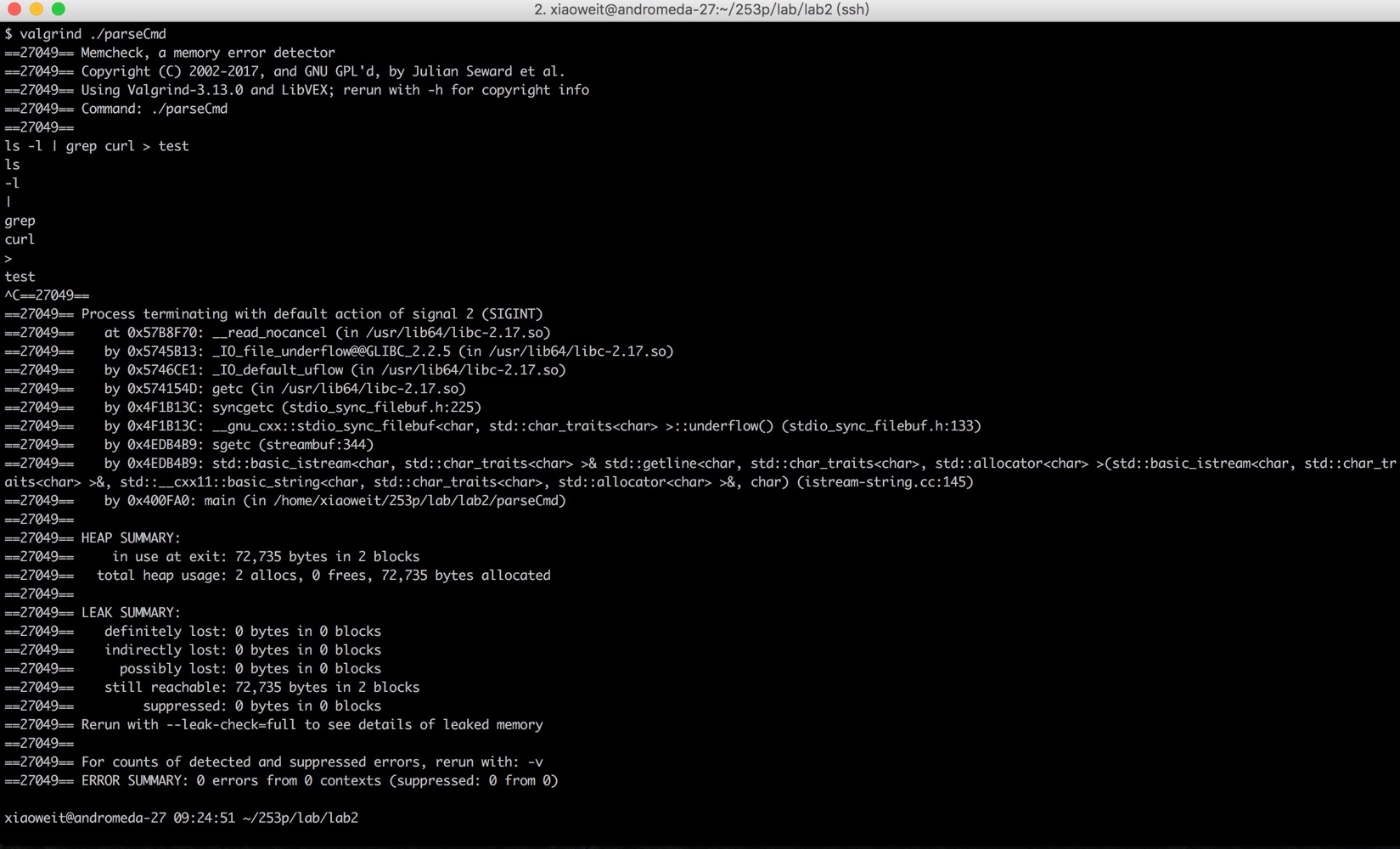
***Advantages:***

It’s a one-pass solution so that it is efficient enough. And also the space complexity is O(1), which means it takes constant extra space.

**Test Cases**

* Testcase1: ./testIntegers < Lab2\_standardizedInput/sampleInts.txt > result
  + output we expect (want)  
    ./testIntegers  
    <  
    Lab2\_standardizedInput/sampleInts.txt  
    >  
    result
  + output our algorithm produces  
    
* Test case 2: git clone ssh://tanxiaowei@gerrit.nh.fangdd.cn:29418/fangdd/xinfang/CustomerMatchServer && scp -p -P 29418 tanxiaowei@gerrit.nh.fangdd.cn:hooks/commit-msg CustomerMatchServer/.git/hooks/
  + output we expect (want)   
    git  
    clone  
    ssh://tanxiaowei@gerrit.nh.fangdd.cn:29418/fangdd/xinfang/CustomerMatchServer  
    &  
    &  
    scp  
    -p  
    -P  
    29418  
    [tanxiaowei@gerrit.nh.fangdd.cn:hooks/commit-msg](mailto:tanxiaowei@gerrit.nh.fangdd.cn:hooks/commit-msg)  
    CustomerMatchServer/.git/hooks/
  + output our algorithm produces   
    
* Test case 3: input from history command
  + output we expect (want)  
    1  
    cd  
    253p/  
    2  
    ls  
    3  
    mkdir  
    lab
  + output our algorithm produces  
    
* Test case 4: standardized test given by instructor
  + output we expect (want)  
    vi  
    foo.cc  
    >  
    output  
    &  
    cat  
    <  
    foo.cc  
    >  
    foo.output  
    &
  + output our algorithm produces  
    

**Screenshot of Compilation and Execution of Program Under Valgrind**

****

**LAB 2, 1/17/2019 MCS 253P**

**Name: Xiaowei Tan**

**Partner: Ethel Hoshi**

**General Problem Description**

Write a function atoi (stands for “ascii” to “integer”) to convert a c-string of base-10 digits to a signed 32 bit decimal number and write a function itoa to convert a signed 32 bit decimal number into a c-string of digits.

**Additional Problem Specifics**

**Sample Input**

Atoi: -123\_frq

Itoa: -123

**Proposed Algorithm**

***Description:***

Atoi：Every time it takes a character into consideration.

* If the current character is space and it’s at the start of the string, then skip it.
* If the character is ‘+’ or ‘-’ after all the spaces, then it knows the sign of the number.

If the character is digit, then add it to the result. If the result is beyond the scope of

integer(-2147483647..2147483647), then output the min(max) value of integer.

* If the character is not digit, terminate the loop and return number according to the sign.

Itoa: At first, confirm the sign of the number. And if it is negative, convert it to its opposite. Then

Every time devide the number by base(2, 8, 10, 16), and store the quotient into the result. And the

assign the remainder to the number. Continue the calculation until number equals zero. Then if

the sign is minus, put a ‘-‘ at the end of the result. Finally, reverse the result string and return it.

***Correctness:***

Atoi starts with the first non-space character, and ends at the first non-digit character. The algorithm simulates the process.

Itoa starts with the last digit of the number and get the result in an inverted order.

***Time Complexity:***

Atoi: O(n)

Itoa: O(n)

***Space Complexity:***

Atoi: O(1)

Itoa: O(n)

**C++ Implementation of Algorithm**

Atoi:

skipSpace(str, index);

sign = getSign(str, index);

num = getNum(str, index, sign);

Itoa:

int sign = getSign(i);

getStr(str, index, base, i);

if(sign < 0)

str[index++] = '-';

str[index] = '\0';

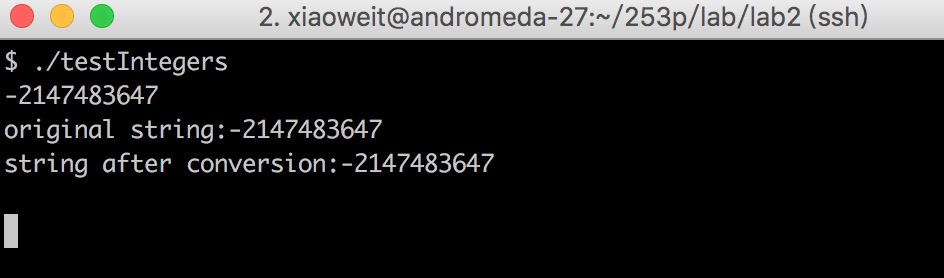
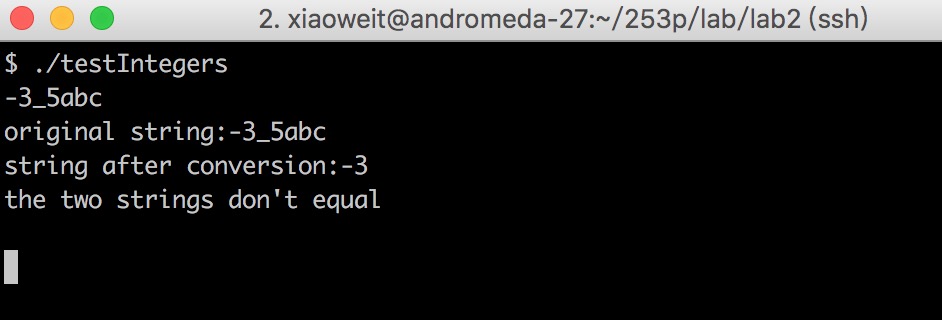
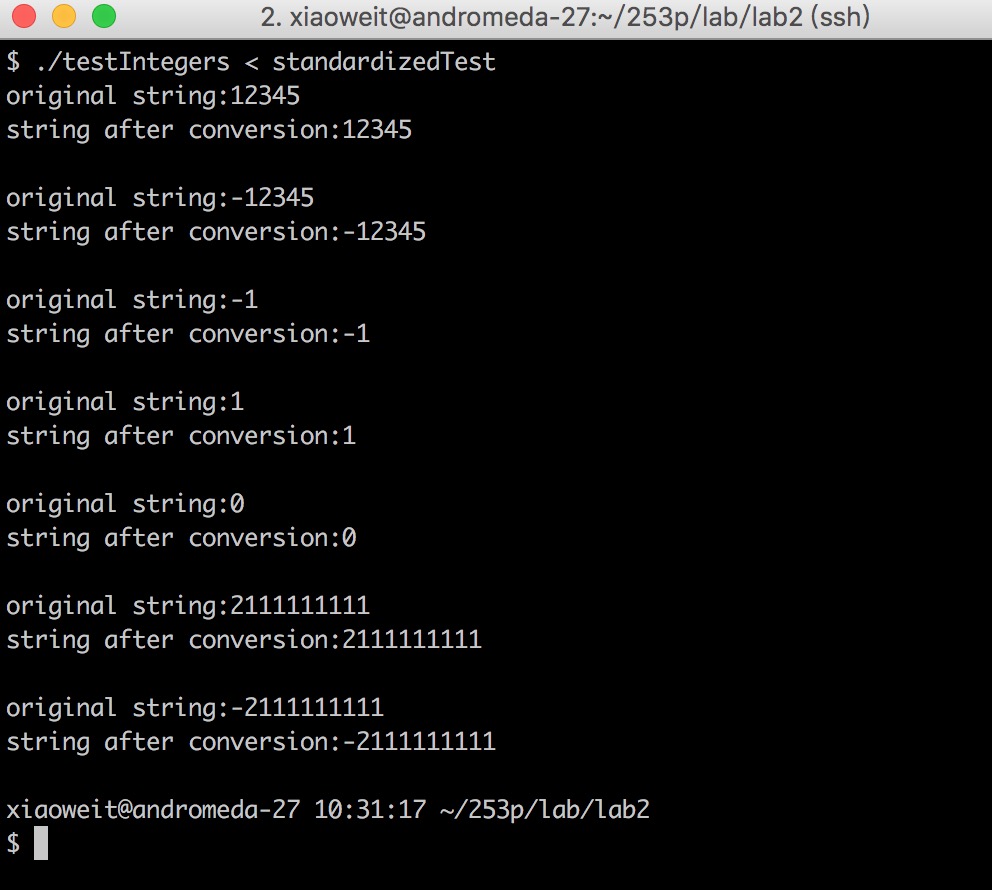
reverse(str, index);

**Advantages/Disadvantages of Your Algorithm and Any Other Comments**

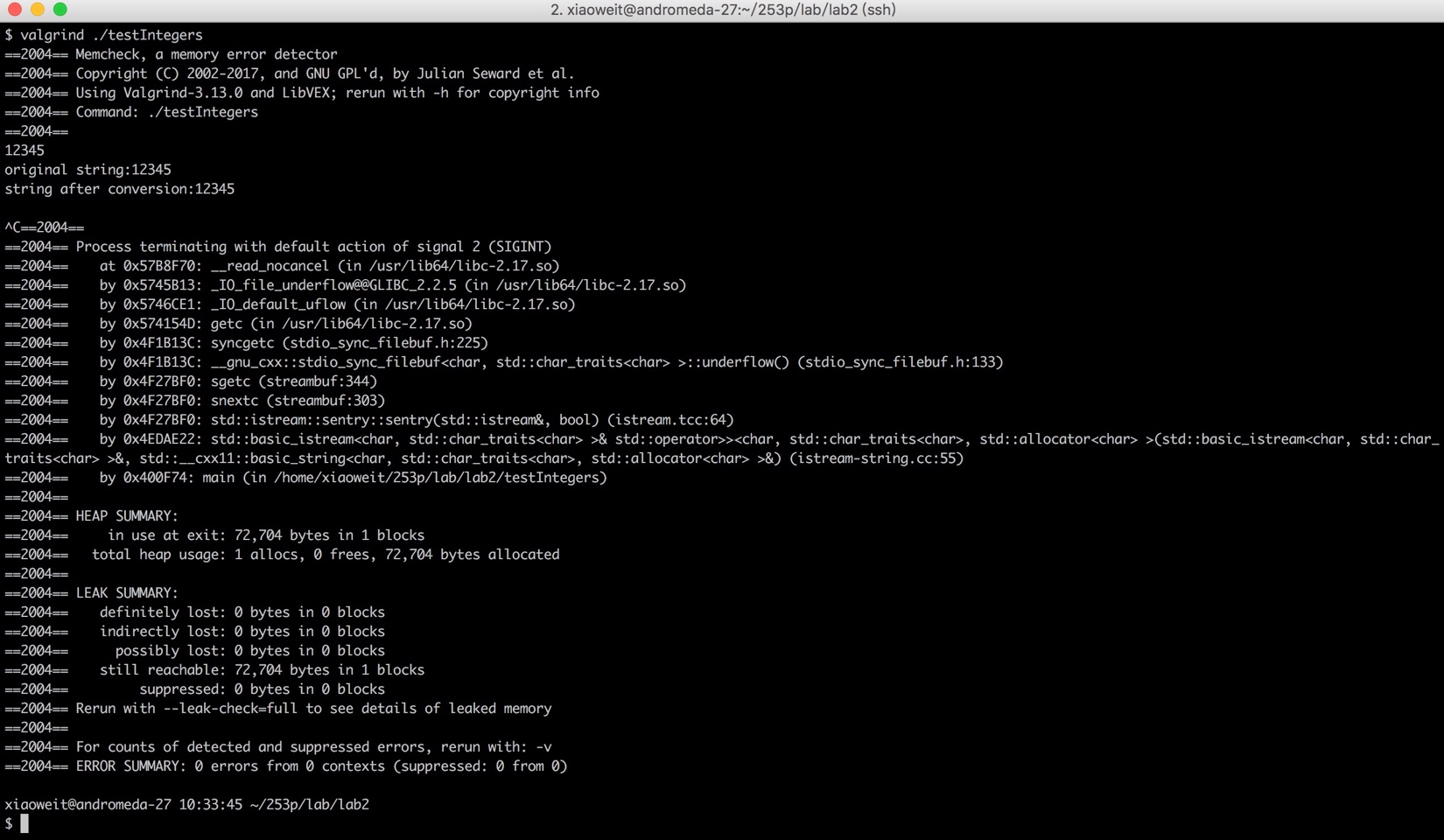
***Advantages:***

Atoi is a one-pass solution and itoa is two-pass. They are both efficient enough and can deal with the corner cases like 2147483647 or -2147483647.

**Test Cases**

* Testcase1: 12345
  + output we expect (want)  
    original string:12345  
    string after conversion:12345
  + output our algorithm produces  
    
* Test case 2: -2147483647
  + output we expect (want)   
    original string:-2147483647  
    string after conversion:-2147483647
  + output our algorithm produces   
    
* Test case 3: -3\_5abc
  + output we expect (want)  
    original string:-3\_5abc  
    string after conversion:-3  
    the two strings don't equal
  + output our algorithm produces  
    
* Test case 4: standardized test given by instructor
  + output we expect (want)  
    original string:12345  
    string after conversion:12345  
    original string:-12345  
    string after conversion:-12345  
    original string:-1  
    string after conversion:-1  
    original string:1  
    string after conversion:1  
    original string:0  
    string after conversion:0  
    original string:2111111111  
    string after conversion:2111111111  
    original string:-2111111111  
    string after conversion:-2111111111
  + output our algorithm produces  
    

**Screenshot of Compilation and Execution of Program Under Valgrind**

****