Biostatistics 615/815 Lecture 19: StatGen Library

Mary Kate Trost
mktrost@umich.edu

March 24, 2011

Outline

- StatGen Library Overview
- Building with the Library
- Useful Classes
  - InputFile
  - String
  - StringArray
  - Parameters
- Documentation
- Debugging

What is the StatGen Library?

- Written by various members of the department
- Contains:
  - C++ library classes
    - General Operation Classes including:
      - File input/output, string processing, parameter parsing
    - Statistical genetics specialized classes
  - Useful Statistical Genetics programs
- Compiles on Linux

Where is the StatGen Library?

- Stored in a git repository
  - https://github.com/statgen/statgen
  - Can browse code & history
- Benefits of a repository:
  - Code consolidated in one location so it is easy to find
  - Bugs can be fixed in one place and picked up by everyone
  - Enable reuse rather than always reinventing
  - Version Tracking
    - If it stops working, you can see what changed & revert

StatGen Library on GitHub

- If your machine has git (preferable) do the following:
  - *cd* <directory where you want the repository located>
  - *git clone https://github.com/statgen/statgen.git*
    - Creates a directory called statgen in the directory where you are located.
  - *cd statgen*
- In the future, update to the latest versions (advantage of using git):
  - *git pull*
  - Recompile from the top level (use a make clean first)
- If you do not have git, follow the instructions on the wiki:
  - http://genome.sph.umich.edu/wiki/File:GithubWithoutGit.pdf

Compiling StatGen Library

- From the statgen directory, build the repository:
  - For normal (optimized) mode, type:
    - *make*
  - For debug, instead use:
    - *make OPTFLAG="-ggdb -O0”*
      - Note: that is the letter O followed by the number zero.
  - When switching between debug & non-debug, prior to typing the new '*make*' call, type:
    - *make clean*

Building with the Library

- Assumes the library has been compiled and exists at: /home/mktrost/statgen (replace with the correct path to your library).
- Create and move into a directory for your code
  - Example: *mkdir ~/statgenDemo; cd ~/statgenDemo*
- Copy the Makefile.
  - *cp /home/mktrost/statgen/src/Makefile.src Makefile*
- .o files will go in an obj directory that will be created

Makefile.tool

- Create Makefile.tool and set the following values:

• EXE = yourExecutableName	Executable name
• TOOLBASE = File1 File2	Basename of files that have both cpp & h
• SRCONLY = Main.cpp	Files with just .cpp and no .h
• HDRONLY = YourTemplate1.h	Files with just .h and no .cpp
• PATH_TO_BASE = /home/mktrost/statgen	Your statgen location
• BIN_DIR = .	Where to put the EXE
• USER_LIBS = /somePath/lib1.a	Any additional libraries
• USER_INCLUDES = -I/somePath/IncDir	Any additional include directories

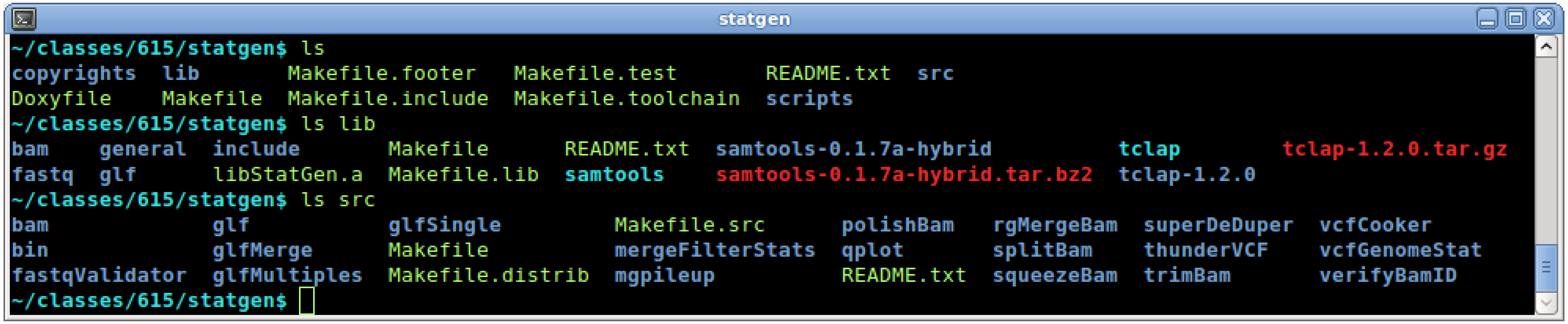
  - Example Makefile.tool:

```
EXE = libraryExample
TOOLBASE = LibraryExample
SRCONLY = Main.cpp
PATH_TO_BASE = /home/mktrost/statgen
BIN_DIR = .
```

- Type *make* or *make OPTFLAG="-ggdb -O0"*

StatGen Library Directory Structure

- copyrights – copyrights for various code pieces
- src – premade statgen programs
- lib – library code
  - **general** – general processing classes ← Focus of this presentation
  - include – library header files are linked here when compiled
  - specialized classes for processing different file types:
    - bam, fastq, glf
  - samtools – samtools software
  - tclap – alternate parameter processing library
  - libStatGen.a – the compiled library



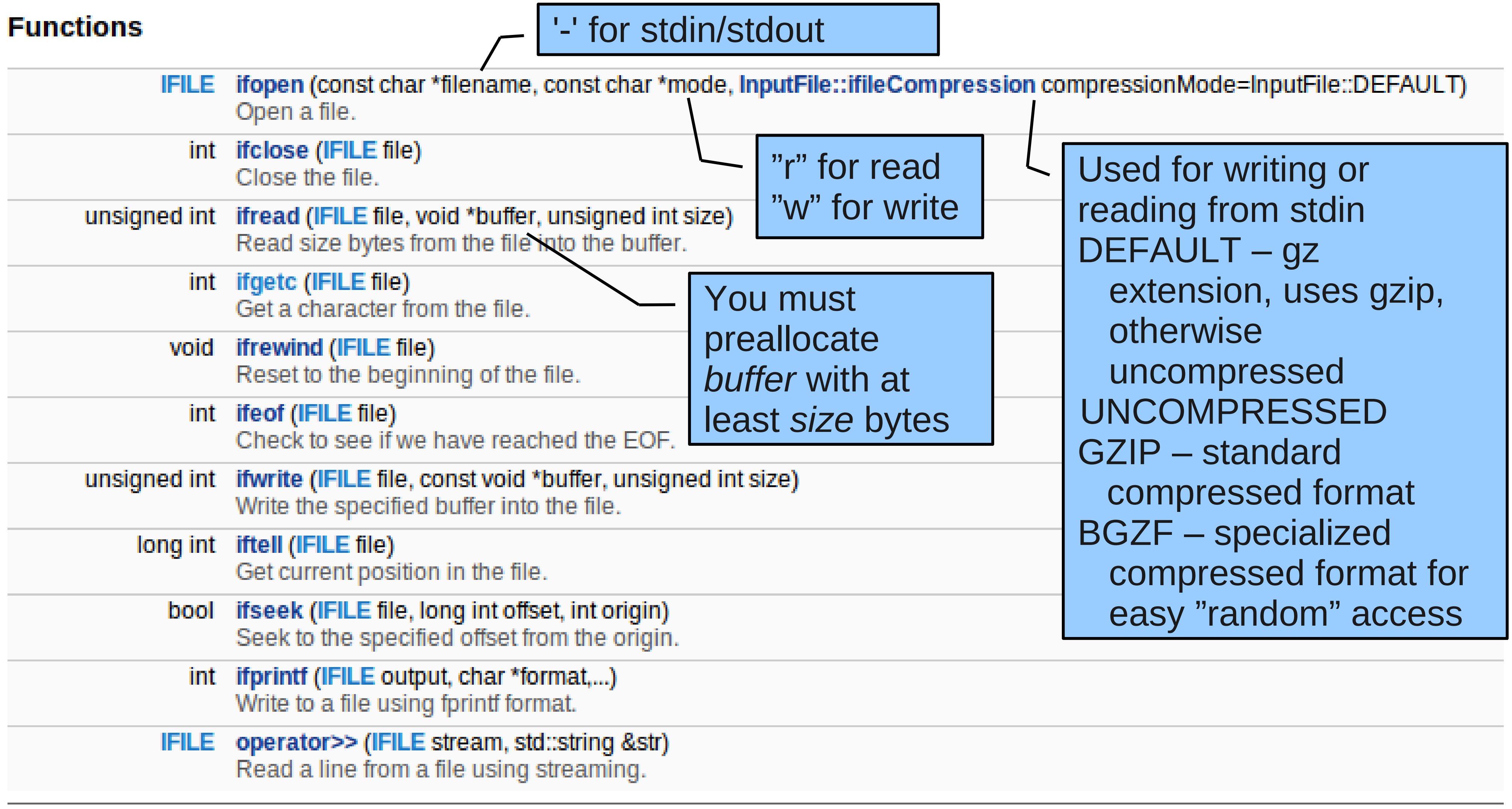
statgen

```
~/classes/615/statgen$ ls
copyrights  lib       Makefile.footer   Makefile.test       README.txt  src
Doxyfile    Makefile  Makefile.include  Makefile.toolchain  scripts
~/classes/615/statgen$ ls lib
bam    general  include       Makefile      README.txt  samtools-0.1.7a-hybrid          tclap        tclap-1.2.0.tar.gz
fastq  glf      libStatGen.a  Makefile.lib  samtools    samtools-0.1.7a-hybrid.tar.bz2  tclap-1.2.0
~/classes/615/statgen$ ls src
bam             glf           glfSingle         Makefile.src      polishBam   rgMergeBam   superDeDuper  vcfCooker
bin             glfMerge      Makefile          mergeFilterStats  qplot       splitBam     thunderVCF    vcfGenomeStat
fastqValidator  glfMultiples  Makefile.distrib  mgpileup          README.txt  squeezeBam   trimBam       verifyBamID
~/classes/615/statgen$
```

InputFile/IFILE

- lib/general/InputFile.h(cpp)
- Read/Write different types of files
    - Uncompressed, BGZF, GZIP
- Reads file to determine type unless reading from stdin
- Flag to specify type for reading stdin and writing
    - Default:
        - gzip for files ending in .gz
        - uncompressed for all others
- IFILE is a pointer to the InputFile class
- Similar to the FILE* interface from the C library

IFILE



Functions

IFILE	**ifopen** (const char *filename, const char *mode, InputFile::ifileCompression compressionMode=InputFile::DEFAULT) Open a file.
int	**ifclose** (IFILE file) Close the file.
unsigned int	**ifread** (IFILE file, void *buffer, unsigned int size) Read size bytes from the file into the buffer.
int	**ifgetc** (IFILE file) Get a character from the file.
void	**ifrewind** (IFILE file) Reset to the beginning of the file.
int	**ifeof** (IFILE file) Check to see if we have reached the EOF.
unsigned int	**ifwrite** (IFILE file, const void *buffer, unsigned int size) Write the specified buffer into the file.
long int	**iftell** (IFILE file) Get current position in the file.
bool	**ifseek** (IFILE file, long int offset, int origin) Seek to the specified offset from the origin.
int	**ifprintf** (IFILE output, char *format,...) Write to a file using fprintf format.
IFILE	**operator>>** (IFILE stream, std::string &str) Read a line from a file using streaming.

'-' for stdin/stdout

"r" for read
"w" for write

You must preallocate *buffer* with at least *size* bytes

Used for writing or reading from stdin
- DEFAULT – gz extension, uses gzip, otherwise uncompressed
- UNCOMPRESSED
- GZIP – standard compressed format
- BGZF – specialized compressed format for easy "random" access

For more detailed descriptions:

http://www.sph.umich.edu/csg/mktrost/doxygen/march22_2011/InputFile_8h.html

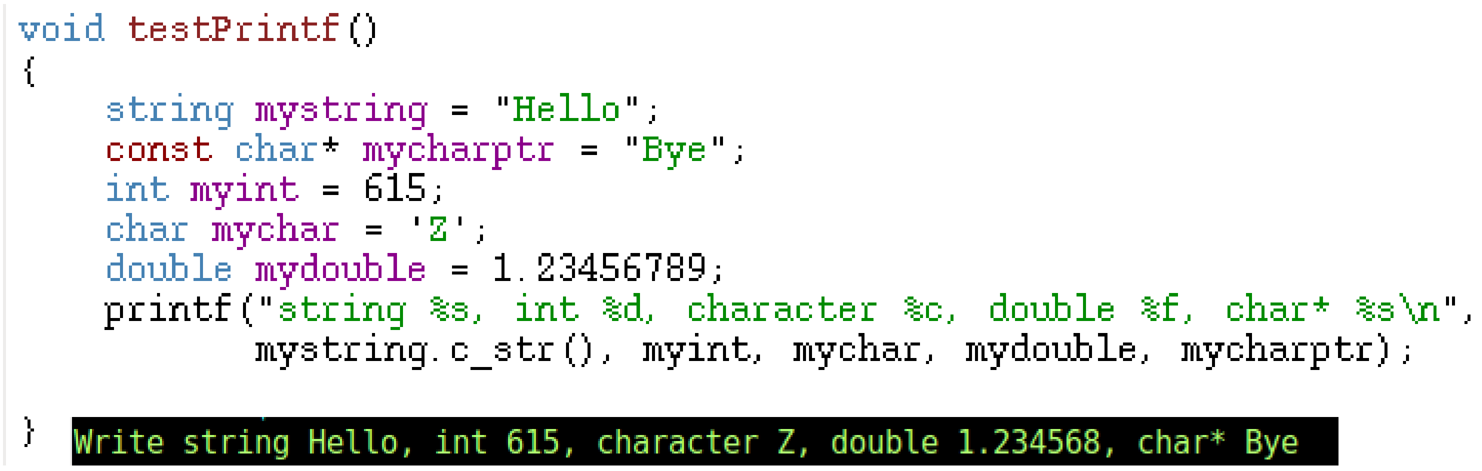
Additional InputFile Methods

Handle buffering of reads.
Low level I/O detail you don't need to worry about

```
/// Set the buffer size for reading from files so that bufferSize bytes
/// are read at a time and stored until accessed by another read call.
/// This improves performance over reading the file small bits at a time.
/// Buffering reads disables the tell call for bgzf files.
/// Any previous values in the buffer will be deleted.
/// \param bufferSize number of bytes to read/buffer at a time,
/// default buffer size is 1048576, and turn off read buffering by setting
/// bufferSize = 1;
void bufferReads(unsigned int bufferSize = DEFAULT_BUFFER_SIZE);

/// Disable read buffering.
void disableBuffering();

/// Returns whether or not the file was successfully opened.
/// \return true if the file is open, false if not.
bool isOpen();

/// Get the filename that is currently opened.
/// \return filename associated with this class
const char* getFileName() const;
```

ifprintf format

- Beware of type safety!
- fprintf format (tons of formatting options):
  - http://www.cplusplus.com/reference/clibrary/cstdio/fprintf/
- printf example:



```
void testPrintf()
{
    string mystring = "Hello";
    const char* mycharptr = "Bye";
    int myint = 615;
    char mychar = 'Z';
    double mydouble = 1.23456789;
    printf("string %s, int %d, character %c, double %f, char* %s\n",
           mystring.c_str(), myint, mychar, mydouble, mycharptr);

}
```

```
Write string Hello, int 615, character Z, double 1.234568, char* Bye
```

- For ifprintf, add your file ptr as the first parameter:
  - ifprintf(myFilePtr, ”int %d”, myint);

Strings

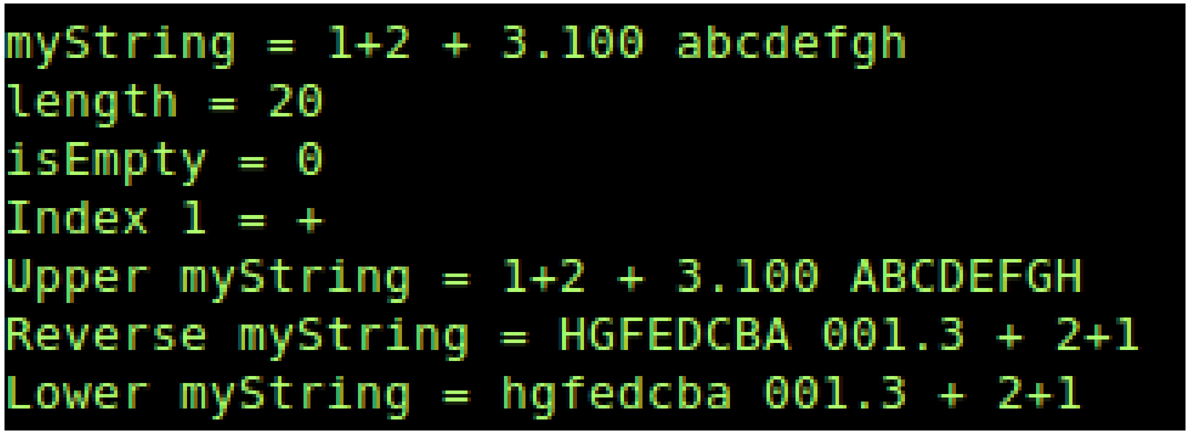
- lib/general/StringBasics.h(cpp)
- Alternative to std::string
  - With advantage of easily adding integers/doubles.
  - Capability of reading a word or line from file.
  - Methods for finding substrings.
- Methods to:
  - *Clear();*
  - *IsEmpty();*
  - *Length();*
  - *ToUpper();*
  - *ToLower();*
  - *Reverse();*
  - *First();*
  - *Last();*
  - *Split(char splitChar) returns vector of strings*
  - *Trim() (from front and back)*

String Operators

- Overloaded operators
  - =, +, +=
    - Can set to or append characters, strings, integers, doubles, and unsigned integers.
    - ==, !=, <, >, >=, <= to other strings.
  - *int*, *double*, *long double* – convert to int, double, long double
  - [ ] - return the character at the index specified in brackets
- Read/Write to file
  - *Read, ReadLine, Write, WriteLine*

String Example

```
void LibraryExample::testString()
{
    String myString;
    int val1 = 2;
    double val2 = 3.1;
    const char* cstring = " abcd";
    String string2 = "efgh";
    myString = 1;
    myString += "+";
    myString += val1;
    myString += " + ";
    myString += val2;
    myString += cstring;
    myString += string2;
    std::cerr << "myString = " << myString << "\n"
              << "length = " << myString.Length() << "\n"
              << "isEmpty = " << myString.IsEmpty() << "\n";
    if(myString.Length() > 1)
    {
        std::cerr << "Index 1 = " << myString[1] << "\n";
    }
    myString.ToUpper();
    std::cerr << "Upper myString = " << myString << "\n";
    myString.Reverse();
    std::cerr << "Reverse myString = " << myString << "\n";
    myString.ToLower();
    std::cerr << "Lower myString = " << myString << "\n";

    myString.Clear();
}
```



```
myString = 1+2 + 3.100 abcdefgh
length = 20
isEmpty = 0
Index 1 = +
Upper myString = 1+2 + 3.100 ABCDEFGH
Reverse myString = HGFEDCBA 001.3 + 2+1
Lower myString = hgfedcba 001.3 + 2+1
```

StringArray

- lib/general/StringArray.h(cpp)
- Useful for tokenizing a string.
  - *int ReplaceColumns(const String & s, char ch = '\t');*
  - *int AddColumns(const String & s, char ch = '\t');*
  - *void Clear();*
  - *int Length() const;*
  - *Index operator [ ]*
- Notes:
  - Read reads the entire file one line into each array entry
  - Write writes each array entry on a separate line.
  - WriteLine writes each array entry on the same line, tab delimited.

Parameters

- One easy way to read parameters
  - lib/Parameters.h(cpp)
- Use for
  - bool, int, double, String
- Long Parameters uses macros & is easy to use
  - Include Parameters.h
  - Declare & initialize the variables you want as parameters
    - Don't forget to initialize otherwise the values are undefined.
  - Set those variables with the type of parameter
  - Read the parameters

Long Parameters

MACRO Name	Parameters	Description
BEGIN_LONG_PARAMETERS	array	Start the parameter declaration creating the variable named array
LONG_PARAMETER_GROUP	label	Create a group named label
LONG_PARAMETER	label, boolptr	Store bool parameter --label in boolptr
LONG_INTPARAMETER	label, intptr	Store int parameter --label in intptr
LONG_DOUBLEPARAMETER	label, doubleptr	Store double parameter --label in doubleptr
LONG_STRINGPARAMETER	label, stringptr	Store String parameter --label in stringptr
END_LONG_PARAMETERS	none	End parameter declaration.

- Each label is the text string that the user should enter for that parameter as --label value
- The ptrs are the address(&) of a previously declared variable.
- Int, double, and String parameters read the next argument as the value.
- Bool parameters are flags that swap true/false each time it appears on the parmeter line
  - if mybool is initialized to false, --mybool –mybool results in false
  - if mybool is initialized to true, --mybool –mybool results in true

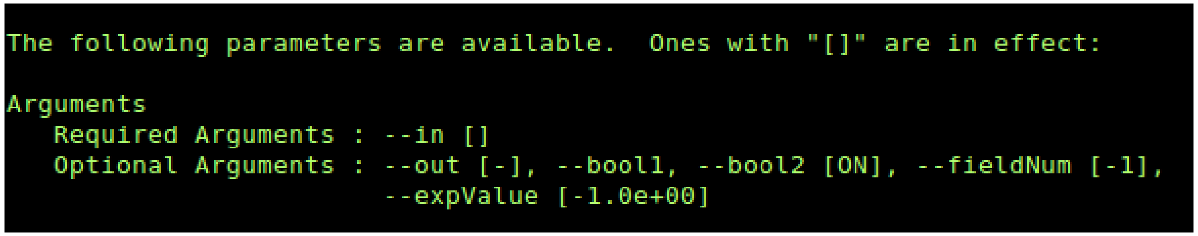
ParameterList

- Declare a ParameterList
  - *ParameterList parmList;*
- Once the LONG_PARAMTERS have been defined, add them to the paramter list.
  - *paramList.Add(new LongParameters(”Arguments”, myParams);*
    - Replace ”Arguments” with the title you want for your parameters
    - Replace myParams with the variable in BEGIN_LONG_PARAMETERS
- Read the parameters from argv
  - *paramList.Read(argc, argv);*
- Optionally print parameter settings
  - *paramList.Status();*

Long Parameters Example

```
//////////////////////////////////////////////
// Create variables to store the parameters
String inFile = ""; // String from statgen/lib/general/
String outFile = "-";
bool bool1 = false; // bool parameter
bool bool2 = true; // bool parameter
int fieldNum = -1; // int parameter
double expValue = -1; // double parameter

// Declare the parameters
BEGIN_LONG_PARAMETERS(longParameterList)
    // using a parameter group is optional.
    LONG_PARAMETER_GROUP("Required Arguments") // 1st parameter group
    LONG_STRINGPARAMETER("in", &inFile)
    LONG_PARAMETER_GROUP("Optional Arguments") // 2nd parameter group
    LONG_STRINGPARAMETER("out", &outFile)
    LONG_PARAMETER("bool1", &bool1)
    LONG_PARAMETER("bool2", &bool2)
    LONG_INTPARAMETER("fieldNum", &fieldNum)
    LONG_DOUBLEPARAMETER("expValue", &expValue)
    // End of the parameters.
    END_LONG_PARAMETERS();

// Create a parameter list
ParameterList paramList;
// Add the parameters to the list.
paramList.Add(new LongParameters ("Arguments",
                                  longParameterList));

// Read the parameters from argv.
paramList.Read(argc, argv);

// Print the parameter status (This step is not necessary).
paramList.Status();
```

Output of program with no arguments specified.



```
The following parameters are available.  Ones with "[]" are in effect:

Arguments
   Required Arguments : --in []
   Optional Arguments : --out [-], --bool1, --bool2 [ON], --fieldNum [-1],
                        --expValue [-1.0e+00]
```

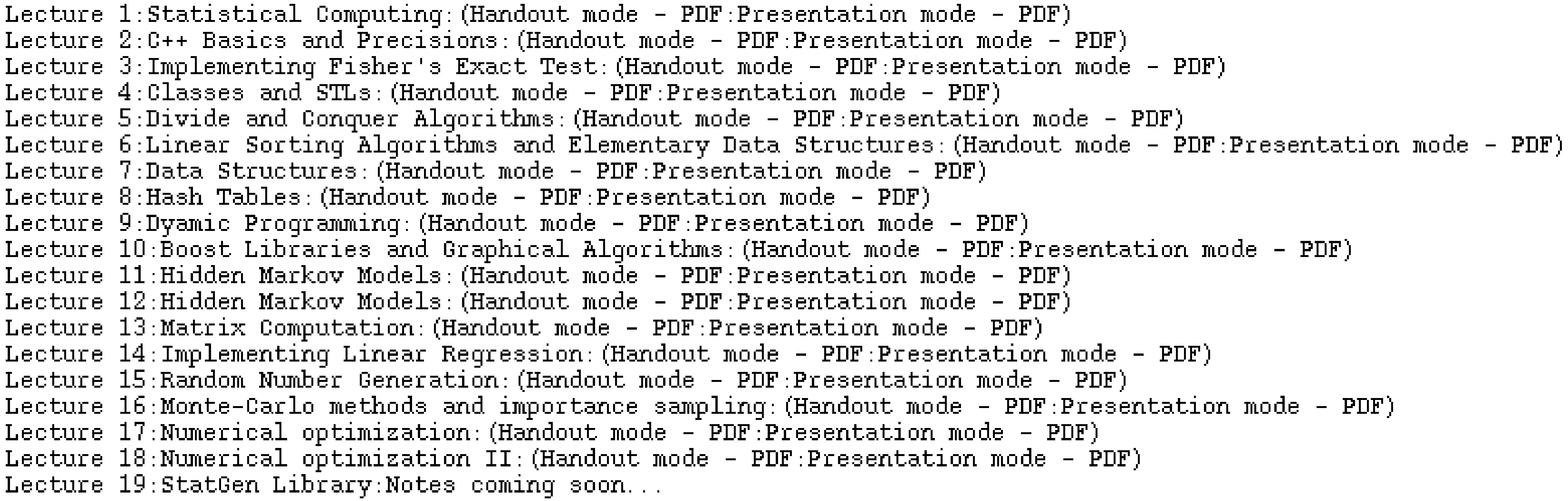
Usage Example

```
void LibraryExample::handleFiles(const String& inFileName, const String& outFileName)
{
    IFILE inputFile = ifopen(inFileName, "r");    // Open the input file.
    IFILE outputFile = ifopen(outFileName, "w"); // Open the output file.

    // Read a line from the file.
    StringArray splitLine;  String currentLine; int bufferSize = 10000; char buffer[bufferSize];
    // String to print at the end of each line.
    const char* endLine = "\tend of line\n"; unsigned int endLineLen = strlen(endLine);

    // Until the end of the input file.
    while(!ifeof(inputFile))
    {
        currentLine.ReadLine(inputFile);  // Read a line using String.
        splitLine.ReplaceColumns(currentLine, ':'); // Tokenize on ':' using StringArray
        // For each column, write the column# & its contents to the output file.
        for(int i = 0; i < splitLine.Length(); i++)
        {
            ifprintf(outputFile, "Col %d: %s\t", i+1, splitLine[i].c_str());
        }
        if(ifwrite(outputFile, endLine, endLineLen) != endLineLen) // Write endLine
        {
            std::cerr << "ERROR writing\n"; // failed to write the correct ammount.
        }
    }

    if(strcmp(inputFile->getFileName(), "-") != 0) // Can't rewind when reading form stdin
    {
        ifrewind(inputFile); // Go back to the beginning.
        int bytesRead = ifread(inputFile, buffer, bufferSize); // Read into buffer.
        // Print the number of bytes read.
        std::cerr << "Number Bytes in File = " << bytesRead << "\n";
    }

    ifclose(inputFile); ifclose(outputFile); // Close the files
}
```

Usage Script Example

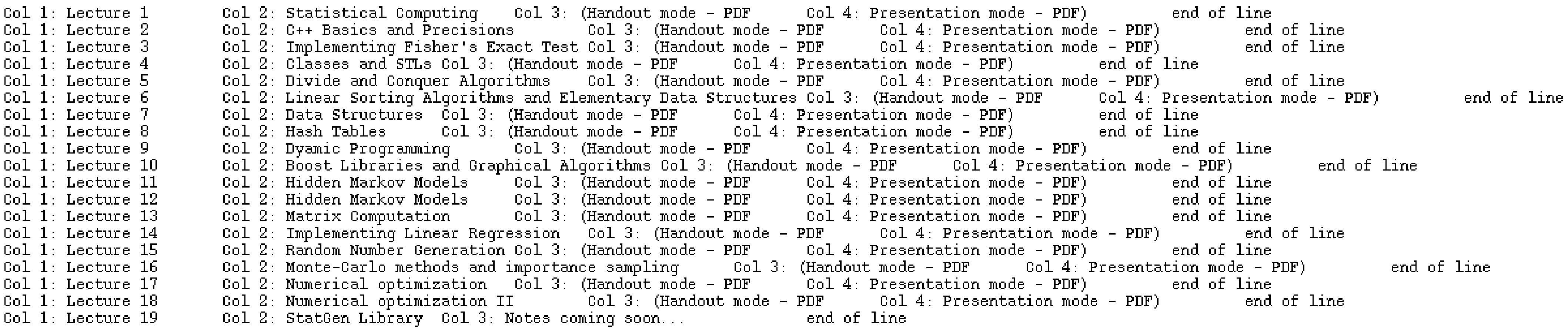
```
mkdir -p out

./libraryExample 2> out/output.txt

# Run the String output
./libraryExample --bool1 2>> out/output.txt

# Run the file processing with text input, text output
./libraryExample --in files/myTestFile.txt --out out/updatedFromTxt.txt 2>> out/output.txt
# Run the file processing with gzip input, text output
./libraryExample --in files/myTestFile.txt.gz --out out/updatedFromGzip.txt 2>> out/output.txt
# verify same results from both types of output
diff out/updatedFromTxt.txt out/updatedFromGzip.txt

# Run the file processing with text input, gzip output
./libraryExample --in files/myTestFile.txt --out out/updatedFromTxtToGz.txt.gz 2>> out/output.txt
# Run the file processing with gzip input, gzip output
./libraryExample --in files/myTestFile.txt.gz --out out/updatedFromGzipToGz.txt.gz 2>> out/output.txt
# verify same results from both types of output
diff out/updatedFromTxtToGz.txt.gz out/updatedFromGzipToGz.txt.gz

# Run the file processing with gzip input, stdout
./libraryExample --in files/myTestFile.txt.gz  2>> out/output.txt | wc
# Run the file processing with gzip input, stdout
./libraryExample --in files/myTestFile.txt.gz  2>> out/output.txt > out/stdOut.txt
# compare stdout to the normal text output
diff out/updatedFromTxt.txt out/stdOut.txt

# Read from stdin
cat files/myTestFile.txt | ./libraryExample --in -  2>> out/output.txt > out/stdIn.txt
# compare stdin to the normal text input
diff out/updatedFromTxt.txt out/stdIn.txt

# compare stderr to expected.
diff expectedOutput.txt out/output.txt
```

Example Input File



```
Lecture 1:Statistical Computing:(Handout mode - PDF:Presentation mode - PDF)
Lecture 2:C++ Basics and Precisions:(Handout mode - PDF:Presentation mode - PDF)
Lecture 3:Implementing Fisher's Exact Test:(Handout mode - PDF:Presentation mode - PDF)
Lecture 4:Classes and STLs:(Handout mode - PDF:Presentation mode - PDF)
Lecture 5:Divide and Conquer Algorithms:(Handout mode - PDF:Presentation mode - PDF)
Lecture 6:Linear Sorting Algorithms and Elementary Data Structures:(Handout mode - PDF:Presentation mode - PDF)
Lecture 7:Data Structures:(Handout mode - PDF:Presentation mode - PDF)
Lecture 8:Hash Tables:(Handout mode - PDF:Presentation mode - PDF)
Lecture 9:Dyamic Programming:(Handout mode - PDF:Presentation mode - PDF)
Lecture 10:Boost Libraries and Graphical Algorithms:(Handout mode - PDF:Presentation mode - PDF)
Lecture 11:Hidden Markov Models:(Handout mode - PDF:Presentation mode - PDF)
Lecture 12:Hidden Markov Models:(Handout mode - PDF:Presentation mode - PDF)
Lecture 13:Matrix Computation:(Handout mode - PDF:Presentation mode - PDF)
Lecture 14:Implementing Linear Regression:(Handout mode - PDF:Presentation mode - PDF)
Lecture 15:Random Number Generation:(Handout mode - PDF:Presentation mode - PDF)
Lecture 16:Monte-Carlo methods and importance sampling:(Handout mode - PDF:Presentation mode - PDF)
Lecture 17:Numerical optimization:(Handout mode - PDF:Presentation mode - PDF)
Lecture 18:Numerical optimization II:(Handout mode - PDF:Presentation mode - PDF)
Lecture 19:StatGen Library:Notes coming soon...
```

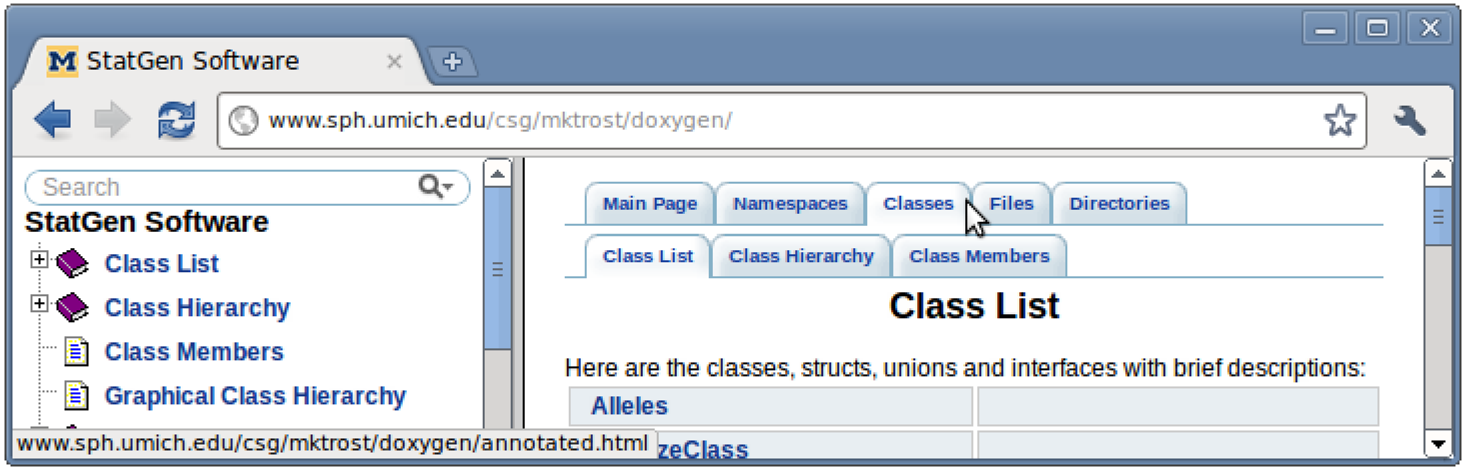
Example Output File



```
Col 1: Lecture 1        Col 2: Statistical Computing    Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 2        Col 2: C++ Basics and Precisions        Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 3        Col 2: Implementing Fisher's Exact Test Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 4        Col 2: Classes and STLs Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 5        Col 2: Divide and Conquer Algorithms    Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 6        Col 2: Linear Sorting Algorithms and Elementary Data Structures Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 7        Col 2: Data Structures  Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 8        Col 2: Hash Tables      Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 9        Col 2: Dyamic Programming       Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 10       Col 2: Boost Libraries and Graphical Algorithms Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 11       Col 2: Hidden Markov Models     Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 12       Col 2: Hidden Markov Models     Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 13       Col 2: Matrix Computation       Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 14       Col 2: Implementing Linear Regression   Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 15       Col 2: Random Number Generation Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 16       Col 2: Monte-Carlo methods and importance sampling      Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 17       Col 2: Numerical optimization   Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 18       Col 2: Numerical optimization II        Col 3: (Handout mode - PDF      Col 4: Presentation mode - PDF)         end of line
Col 1: Lecture 19       Col 2: StatGen Library  Col 3: Notes coming soon...             end of line
```

Library Documentation

- Wiki pages:
  - For all software, library & tools:
    - http://genome.sph.umich.edu/wiki/Software
  - Library specific documentation:
    - http://genome.sph.umich.edu/wiki/C%2B%2B_Library:_libStatGen
- Doxygen
  - http://www.sph.umich.edu/csg/mktrost/doxygen/march22_2011/
  - For list of Classes:
    - Select "Classes"



Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Alleles

zeClass

www.sph.umich.edu/csg/mktrost/doxygen/annotated.html