Programming in C/C++
Exercises set six: Basic Input/Output

Christiaan Steenkist
Diego Ribas Gomes
Jaime Betancor Valado
Remco Bos

October 19, 2016

Exercise 50, creating a 'one-size-fits-all' function

The exercise gives a main function that won't compile because it lacks a function declaration. The declaration must allow for different types and number of arguments and is added in the following source.

Listing 1: variadic.cc

```
1 void fun(...){}
2
3 int main()
4 {
5     fun();
6     fun("with functions");
7     fun(1, 2, 3);
8 }
```

Exercise 51, understanding the behaviour of istreams

A piece of code is presented which is expected to take and output two numerical unsigned inputs, but outputs 0 instead of the second value. The reason for this is that the stream's data hasn't been activated by a seek operation after receiving the value by the << operator. The fix for this behaviour is added to the code.

Listing 2: Code repair in line 3

```
1     cout << "extracted first number: " << no1 << '\n';
2
3     istr.seekg(0);
4
5     istr.str(argv[2]);
6     size_t no2 = 0;
7     istr >> no2;
```

Exercise 52, defining a manipulator

A manipulator is implemented to insert the current date and time as produced by asctime but without the trailing newline it automatically appends.

Listing 3: nowManip.cc

```
1  #include <iostream>
2  #include <ctime>
3  #include <cstring>
4
5  using namespace std;
6
7  ostream &now(ostream &stream)
8  {
9    struct tm *newTime;
10   time_t ltime;
11
12   time(&ltime);
13
14   newTime = localtime(&ltime);
15
16   char *currentTime = asctime(newTime);
17   currentTime[strlen(currentTime) - 1] = '\0';
18
19   return stream << currentTime;
20 }
21
22 int main()
23 {
24   cout << now << '\n';
25 }
```

Exercise 53, displaying floating point numbers using modifiers

A program is built that defines a double variable and displays it in different requested formats using a single cout statement.

Listing 4: floatModif.cc

```
1  #include <iostream>
2  #include <iomanip>
3
4  using namespace std;
5
6  int main()
7  {
8      double value = 12.04;
```

```
9
10      cout
11      << setw(15) << value << "\n"
12      << setw(15) << left << value << "\n"
13      << setw(15) << right << value << "\n"
14      << setw(15) << fixed << setprecision(1) << value << "\n"
15      << setw(15) << setprecision(4) << value << "\n"
16      << resetiosflags(ios::fixed) << setw(15) << value << "\n";
17  }
```

Exercise 54, using binary files

A program is built reading by default the binary file /var/log/account/pacct or another specified command-line argument and outputs the names of all processes that didn't exit properly. The inclusion of a '-a' option should make it output information about all exited processes and if a process was killed with SIGKILL or SIGTERM, it should mention the name of the signal instead of the number.

Listing 5: main.cc

```
1  #include "main.h"
2
3  int main(int argc, char **argv)
4  {
5    Vars vars;
6    arguments(vars, argc, argv);
7    process(vars);
8  }
```

Listing 6: main.h

```
1  #ifndef MAIN_H_
2  #define MAIN_H_
3
4  #include<iostream>
5
6  enum OPTION
7  {
8    ABRIDGED, VERBOSE
9  };
10
11 struct Vars
12 {
13   OPTION method = ABRIDGED;
14   std::string path = "/var/log/account/pacct";
15 };
16
17 void arguments(Vars &vars, int argc, char **argv);
18 void process(Vars &vars);
```

```
19 void printProcess(char *name, size_t exitCode);
20
21 #endif
```

Listing 7: main.ih

```
1  #include "main.h"
2
3  #include <iostream>
4  #include <fstream>
5  #include <unistd.h>
6  #include </usr/include/linux/acct.h>
7  #include <csignal>
8
9  using namespace std;
```

Listing 8: arguments.cc

```
1  #include "main.ih"
2
3  void arguments(Vars &vars, int argc, char **argv)
4  {
5    // If there are no arguments do nothing.
6    if (argc == 1)
7      return;
8
9    // If an option -a is found then the method is set to verbose.
10   int opt;
11   while((opt = getopt(argc, argv, "a")) != EOF)
12   {
13     switch (opt)
14         {
15       case 'a':
16         vars.method = VERBOSE;
17       break;
18       default:
19         cerr << "Invalid option " << opt << '\n';
20         return;
21       break;
22     }
23   }
24
25   if (vars.method == VERBOSE)
26   {
27     if (argc >= 3)
28       vars.path = argv[2];
29   }
30   else
```

```
31      vars.path = argv[1];
32  }
```

Listing 9: process.cc

```
1   #include "main.ih"
2
3   // Size of the acct_v3 struct
4   // as well as relative positions
5   // of the name and exit code.
6   size_t stepSize = sizeof(acct_v3);
7   size_t relNamePos = stepSize -
8             sizeof(char) * ACCT_COMM;
9   size_t relCodePos = stepSize +
10            sizeof(char) * 2 +
11            sizeof(__u16);
12
13  void process(Vars &vars)
14  {
15    ifstream is(vars.path, ifstream::binary);
16
17    if (!is)
18    {
19      cerr << "No file found.\n";
20      cerr << vars.path << '\n';
21      return;
22    }
23
24    // Get the length of the file and
25    // return to the beginning.
26    is.seekg(0, is.end);
27    size_t length = is.tellg();
28    is.seekg(0, is.beg);
29
30    char name[ACCT_COMM];
31    __u32 exitCode;
32
33    size_t process = 0;
34    size_t place = 0;
35    while ((place = process * stepSize) < length)
36    {
37      is.seekg(place + relCodePos, is.beg);
38      is.read(reinterpret_cast<char *>(&exitCode), sizeof(exitCode));
39
40      is.seekg(place + relNamePos, is.beg);
41      is.read(reinterpret_cast<char *>(&name), sizeof(char) *
       ACCT_COMM);
42
```

```
43      if (exitCode != 0 || vars.method == VERBOSE)
44        printProcess(name, exitCode);
45
46      ++process;
47    }
48  }
```

Listing 10: printProcess.cc

```
1  #include "main.ih"
2
3  void printProcess(char *name, size_t exitCode)
4  {
5    cout << '\'' << name << "\' ";
6    switch(exitCode)
7    {
8      case(SIGKILL):
9        cout << "KILL";
10     break;
11     case(SIGTERM):
12       cout << "TERM";
13     break;
14     default:
15       cout << exitCode;
16     break;
17   }
18   cout << '\n';
19 }
```