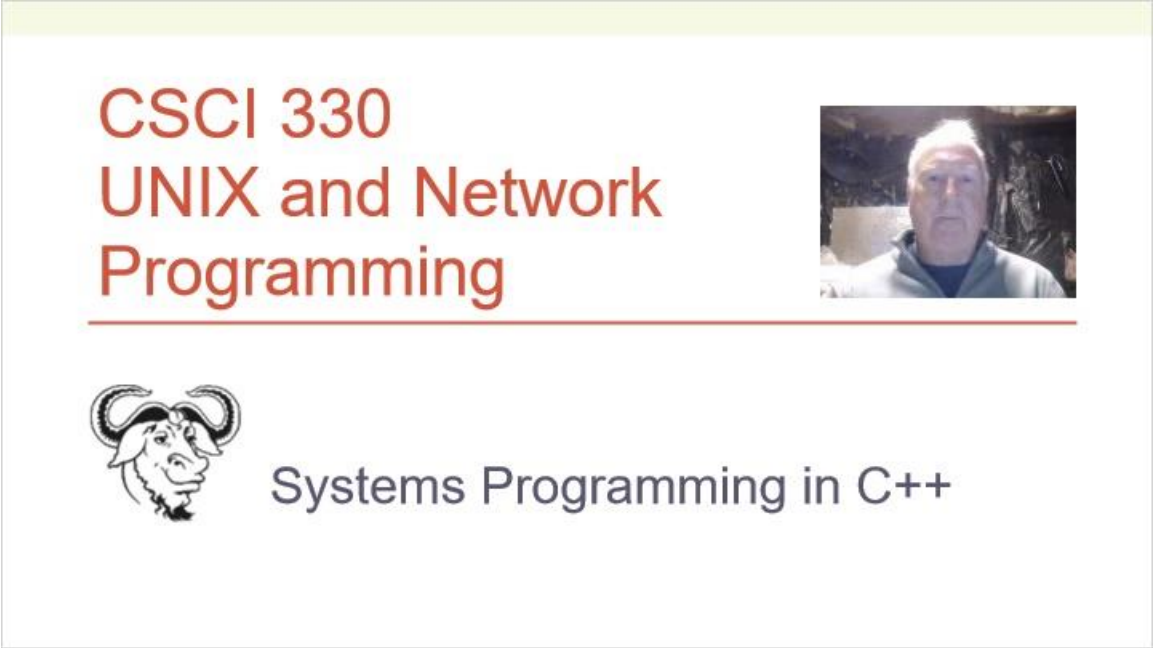
Systems Programming in C++

1. Systems Programming

1.1 CSCI 330



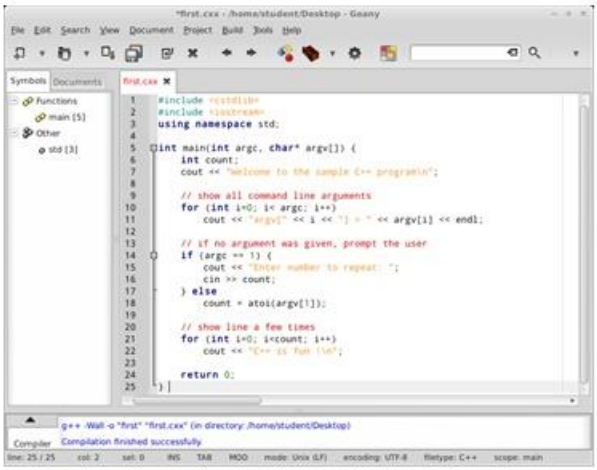
1.2 Unit Overview

Unit Overview

- C++ review
  - programming with Geany
- C Library functions
  - C strings
  - environment access
  - regular expressions
  - directory I/O

1.3 C++ Review

C++ Review



1.4 C Library Functions

C Library Functions

- C library is accessible to C++
- I/O operations: `#include <cstdio>`
  - functions: printf, scanf, putchar, getchar, ...
- Strings: `#include <cstring>`
  - functions: strlen, strcat, strcpy, strstr, memset, ...
- Standard general utilities: `#include <cstdlib>`
  - functions: atoi, rand, malloc, free, getenv, exit, system, ...

- Reference: http://www.cplusplus.com/reference/clibrary

1.5 C Strings: strcpy

C Strings: strcpy

```
char* strcpy(char* dest, const char* source)
```

- copies the C string pointed to by **source** into the array pointed to by **dest**, including the terminating *null* character

- to avoid overflows, the size of the array pointed to by **dest** must be long enough to contain the same C string as **source**, including the terminating *null* character

1.6 C Strings: strcpy

C Strings: strcpy

strcpy.cxx

```
/*
 * strcpy.cxx
 *
 *  Example Program for CSC
 *  demonstrates string cop
 *
 */
#include <cstring>
#include <iostream>
using namespace std;

int main () {
  char str1[]="Sample string";
  char str2[20];
  char str3[20];
  strcpy (str2,str1);
  strcpy (str3,"copy successful");
  cout << "str1: " << str1 << endl;
  cout << "str2: " << str2 << endl;
  cout << "str3: " << str3 << endl;
  return 0;
}
```

Terminal - student@csci330: ~/Desktop/Week 9 - Systems Programming

File Edit View Terminal Tabs Help

```
student@csci330:~/Desktop/Week 9 - Systems Programming$ ./strcpy
str1: Sample string
str2: Sample string
str3: copy successful
student@csci330:~/Desktop/Week 9 - Systems Programming$
```

1.7 C Strings: strlen, strcmp

C Strings: strlen, strcmp

`int strlen(char *s)`

- calculate the length of string **`s`**

  (not counting the terminating *null*)

`int strcmp(char *s1, char *s2)`

- compares the two strings **`s1`** and **`s2`**, returns 0 if equal

1.8 C Strings: strlen

C Strings: strlen

strlen.cxx

```
/*
 * strlen.cxx
 *
 *  Example Program for CSCI 330
 *  demonstrates string library functions
 *
 */
#include <cstring>
#include <iostream>
using namespace std;

int main () {
   char str1[]="Sample string";
   char str2[20];
   char str3[20];
   strcpy (str2,str1);
   strcpy (str3,"copy successful");
   cout << "'" << str1 << "' is " << strlen(str1) << " characters long\n";
   cout << "'" << str2 << "' is " << strlen(str2) << " characters long\n";
   cout << "'" << str3 << "' is " << strlen(str3) << " characters long\n";
   return 0;
}
```

Terminal -

File Edit View Terminal Tabs Help

```
'Sample string' is 13 characters long
'Sample string' is 13 characters long
'copy successful' is 15 characters long
```

1.9 C Strings: strcmp

C Strings: strcmp

strcmp.cxx

```
/*
 * strcmp.cxx
 *
 *  Example Program for CSCI 330
 *  demonstrates string library functions
 *
 */
#include <cstring>
#include <iostream>
using namespace std;

int main () {
   char str1[]="Sample string";
   char str2[20];
   char str3[20];
   strcpy (str2,str1);
   strcpy (str3,"copy successful");
   if (strcmp(str1, str2) == 0) {
      cout << "'" << str1 << "' and '" << str2 << "' are the same\n";
   }
   if (strcmp(str1, str3) != 0) {
      cout << "'" << str1 << "' and '" << str3 << "' are not the same\n";
   }
   return 0;
}
```

Terminal -

File Edit View Terminal Tabs Help

```
'Sample string' and 'Sample string' are the same
'Sample string' and 'copy successful' are not the same
```

1.10 C Strings: strtok

C Strings: strtok

`char *strtok(char *str, char *delim)`

- extract tokens from string **`str`**, delimited by **`delim`**
- if **`str`** is **`NULL`** then next token in previous **`str`** is foun
- returns **`NULL`** if there no more tokens in **`str`**
- example delimiter strings:

  `" "` `" .,:;!?"`

1.11 C Strings: strtok

C Strings: strtok

strtok.cxx

```
/*
 * strtok.cxx
 *
 *  Example Program for CSCI 330
 *  demonstrates string manipulation functions
 *
 */
#include <cstring>
#include <iostream>
using namespace std;

int main () {
  char text[]="The quick brown fox jumps over the lazy dog";
  cout << text << "\nis " << strlen(text) << " characters long.\n";
  // get indvidual words from text
  char *first = strtok(text, " ");
  cout << "1st word: " << first << endl;
  char *second = strtok(NULL, " ");
  cout << "2nd word: " << second << endl;
  // loop to get the remaining words
  char *next;
  while (next=strtok(NULL, " ")) {
    cout << "next word: " << next << endl;
  }
  return 0;
}
```

Terminal -

File Edit View Terminal Tabs Help

```
The quick brown fox jumps over the lazy dog
is 43 characters long.
1st word: The
2nd word: quick
next word: brown
next word: fox
next word: jumps
next word: over
next word: the
next word: lazy
next word: dog
```

1.12 General Utility: getenv

General Utility: getenv

```
char * getenv ( const char * name )
```

- gets environment string
- retrieves a C string containing the value of the environment variable whose name is specified as argument
- if the requested variable is not part of the environment list, the function returns a null pointer

1.13 General Utility: getenv

General Utility: getenv

getenv.cxx

```
1  /*
2   * getenv.cxx
3   *
4   *  Example Pro
5   *  demonstrate
6   *
7   */
8  #include <iostr
9  #include <cstdlib>
10 using namespace std;
11
12 int main () {
13    char * path;
14    path = getenv ("PATH");
15    if (path != NULL)
16       cout << "The current path is: " << path << endl;
17    return 0;
18 }
19
```

Terminal - student@csci330: ~/Desktop/Week 9 - Systems Programming

File Edit View Terminal Tabs Help

```
student@csci330:~/Desktop/Week 9 - Systems Programming$ ./getenv
The current path is: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bi
n:/usr/games:/usr/local/games
student@csci330:~/Desktop/Week 9 - Systems Programming$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loc
al/games
student@csci330:~/Desktop/Week 9 - Systems Programming$
```

1.14 General Utility: exit

General Utility: exit

```
void exit ( int status )
```

- terminates calling process
- if status is zero or EXIT_SUCCESS, a successful termination status is returned to the host environment
- if status is EXIT_FAILURE, an unsuccessful termination status is returned to the host environment

1.15 General Utility: system

General Utility: system

```
int system ( const char * command )
```

- invokes the command processor, i.e. shell, to execute a command
- returns exit status of command
- if command is a null pointer, the function only checks whether a command processor is available

1.16 General Utility: system

General Utility: system

system.cxx

```
1   /*
2    * system.cxx
3    *
4    *  Example Program for CSCI 330
5    *  demonstrates system and exit library function
6    *
7    */
8   #include <iostream>
9   #include <cstdlib>
10  #include <cstdio>
11  using namespace std;
12
13  int main () {
14     int rs;
15     cout << "Checking if processor is available ...
16     if (system(NULL))
17        cout << " Ok \n";
18     else
19        exit (EXIT_FAILURE);
20     cout << "Executing command ls -l ...\n";
21     rs = system ("ls -l");
22     return rs;
23  }
24
```

Terminal - student@csci330: ~/Desktop/Week 9 - Systems Programming

File Edit View Terminal Tabs Help

```
student@csci330:~/Desktop/Week 9 - Systems Programming$ ./system
Checking if processor is available ... Ok
Executing command ls -l ...
total 140
-rwxrwxr-x 1 student student 13256 Feb 21 12:04 first
-rw-rw-r-- 1 student student   595 Feb 21 12:08 first.cxx
-rw-rw-r-- 1 student student  3704 Feb 21 12:04 first.o
-rwxrwxr-x 1 student student  8968 Feb 21 13:44 getenv
-rw-rw-r-- 1 student student   328 Feb 14 16:52 getenv.cxx
-rw-rw-r-- 1 student student   879 Feb 19 13:14 oter.cxx
-rw-rw-r-- 1 student student   608 Feb 14 16:52 readDir.cxx
-rw-rw-r-- 1 student student   736 Feb 14 16:52 regex.cxx
-rwxrwxr-x 1 student student  9024 Feb 21 13:30 strcpy
-rw-rw-r-- 1 student student   427 Feb 14 16:52 strcpy.cxx
-rw-rw-r-- 1 student student  3472 Feb 21 13:30 strcpy.o
-rwxrwxr-x 1 student student  8840 Feb 26 12:17 system
-rw-rw-r-- 1 student student   429 Feb 14 16:52 system.cxx
student@csci330:~/Desktop/Week 9 - Systems Programming$
```

1.17 Regular Expression Library Functions

Regular Expression Library Functions

- allows to use regular expression to search strings

<u>functions:</u>

regcomp to prepare, i.e. compile, a regular expression

regexec to use prepared expression to search a string

1.18 C Library: Regex

C Library: Regex

```
Terminal
File  Edit  View  Search  Terminal  Help
REGEX(3)                Linux Programmer's Manual                REGEX(3)

NAME
       regcomp, regexec, regerror, regfree - POSIX regex functions

SYNOPSIS
       #include <sys/types.h>
       #include <regex.h>

       int regcomp(regex_t *preg, const char *regex, int cflags);

       int regexec(const regex_t *preg, const char *string, size_t nmatch,
                   regmatch_t pmatch[], int eflags);

       size_t regerror(int errcode, const regex_t *preg, char *errbuf,
                       size_t errbuf_size);

       void regfree(regex_t *preg);

DESCRIPTION
   POSIX Regex Compiling
       regcomp() is used to compile a regular expression into a form that is
       suitable for subsequent regexec() searches.

 Manual page regcomp(3) line 1 (press h for help or q to quit)
```

1.19 Library Function: regcomp

Library Function: regcomp

`int regcomp(regex_t *preg, const char *regex, int cflags)`

- compiles the regular expression string **`regex`** into a **`regex_t`** structure **`preg`** for later use
- **`cflags`** allows variations to regular expression styles
  - 0 selected basic regular expression syntax
  - REG_EXTENDED uses extended regular expression syntax

1.20 Library Function: regexec

Library Function: regexec

```
int regexec(const regex_t *preg, const char *string,
            size_t nmatch, regmatch_t pmatch[], int eflags)
```

- uses regular expression **preg** to analyze **string**
- **nmatch** and **pmatch[]** return the location of matches
  - can be set to 0 if not needed
- **eflags** allows variations on end of line matching
  - can be set to 0 if not needed
- returns 0 for successful match

1.21 regex.cxx

regex.cxx

```
/*
 * regex.cxx
 *
 *  Example Program for CSCI 330
 *  shows regcomp and regexec library functions
 *
 */
#include <sys/types.h>
#include <regex.h>
#include <unistd.h>
#include <cstdlib>
#include <cstdio>
#include <iostream>
using namespace std;

int main(int argc, char* argv[]) {
    // check for arguments
    if (argc < 3) {
        cerr << "usage: regex text regex";
        exit(EXIT_FAILURE);
    }
    cout << "Search for: " << argv[2] << " in " << argv[1] << endl;

    regex_t regex;
    int rs;

    // prepare regular expression
    regcomp(&regex, argv[2], 0);

    // execute the regular expression
    rs = regexec(&regex, argv[1], 0, 0, 0);
    if (rs == 0)
        cout << argv[1] << " is matched by: " << argv[2] << endl;
    else
        cout << "no match" << endl;
}
```

line: 1 / 37 col: 0 sel: 0 INS TAB mode: LF encoding: UTF-8 filetype: C++ scope: unknown

1.22 Error Handling

Error Handling

- convention on how to report errors
  - return -1 in return status
  - set global variable **errno**
  - **errno** is index into table of error messages
- C library function **perror** translates this error code and prints understandable error message

1.23 C Library Function: perror

C Library Function: perror

```
Terminal
File  Edit  View  Search  Terminal  Help
PERROR(3)                Linux Programmer's Manual                PERROR(3)

NAME
       perror - print a system error message

SYNOPSIS
       #include <stdio.h>

       void perror(const char *s);

       #include <errno.h>

       const char *sys_errlist[];
       int sys_nerr;
       int errno;

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       sys_errlist, sys_nerr: _BSD_SOURCE

DESCRIPTION
       The routine perror() produces a message on the standard error output,
       describing the last error encountered during a call to  a  system  or
       library  function.  First (if s is not NULL and *s is not a null byte
       ('\0')) the argument string s is printed, followed by a colon  and  a
       blank.  Then the message and a new-line.
 Manual page perror(3) line 1 (press h for help or q to quit)
```

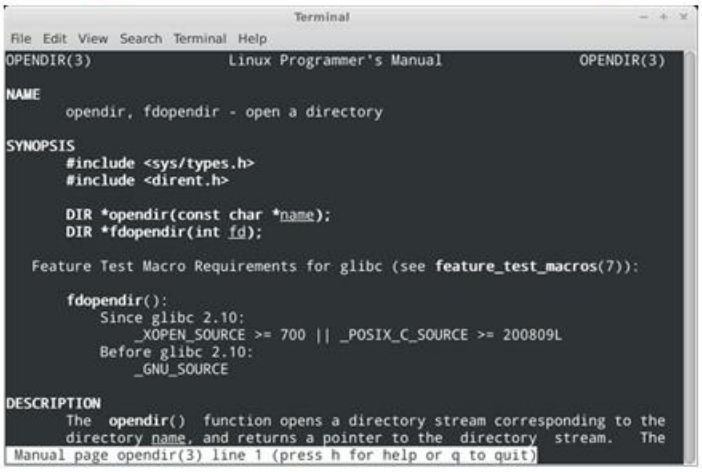
1.24 Directory Input/Output

Directory Input/Output

- current directory: **`chdir`**, **`getcwd`**
- directory I/O functions: **`opendir`**, **`readdir`**
- directory I/O types: **`DIR`**, **`struct dirent`**

1.25 Directory I/O function: opendir

Directory I/O function: opendir



1.26 Directory I/O function: opendir

Directory I/O function: opendir

```
DIR *opendir(const char *name)
```

- opens directory **name** as a stream
- returns **DIR** pointer for **readdir** function
- returns NULL on error, and **errno** is:
  - **ENOENT** directory does not exist
  - **ENOTDIR** name is not a directory

1.27 Directory I/O function: readdir

Directory I/O function: readdir

```
Terminal
File  Edit  View  Search  Terminal  Help
READDIR(3)                Linux Programmer's Manual                READDIR(3)

NAME
       readdir, readdir_r - read a directory

SYNOPSIS
       #include <dirent.h>

       struct dirent *readdir(DIR *dirp);

       int readdir_r(DIR *dirp, struct dirent *entry, struct dirent **result);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       readdir_r():
           _POSIX_C_SOURCE >= 1 || _XOPEN_SOURCE || _BSD_SOURCE ||
           _SVID_SOURCE || _POSIX_SOURCE

DESCRIPTION
       The readdir() function returns a pointer to a dirent structure  repre-
       senting the next directory entry in the directory stream pointed to by
       dirp.  It returns NULL on reaching the end of the directory stream  or
       if an error occurred.
 Manual page readdir(3) line 1 (press h for help or q to quit)
```

1.28 Directory I/O function: readdir

Directory I/O function: readdir

```
struct dirent *readdir(DIR *dirp)
```

- returns a pointer to a **dirent** structure representing the next directory entry in directory **dirp**
- returns NULL on reaching end of directory or if an error occurred

1.29 dirent structure

dirent structure

```
struct dirent {
  ino_t          d_ino;       /* inode number */
  off_t          d_off;       /* offset to next */
  unsigned short d_reclen;    /* record length */
  unsigned char  d_type;      /* type of file */
  char           d_name[256]; /* filename */
};
```

1.30 Illustration: directory listing

Illustration: directory listing

readDir.cxx ×

```
7   #include <dirent.h>
8   #include <cstdio>
9   #include <cstdlib>
10  #include <iostream>
11  using namespace std;
12
13  int main(int argc, char* argv[]) {
14      if (argc != 2) {
15          cerr << "USAGE: readDir pathname\n";
16          exit(EXIT_FAILURE);
17      }
18
19      DIR *dirp;
20      struct dirent *dirEntry;
21
22      // open directory
23      dirp = opendir(argv[1]);
24      if (dirp == 0) {
25          perror(argv[1]);
26          exit(EXIT_FAILURE);
27      }
28
29      while ((dirEntry = readdir(dirp)) != NULL) {
30          cout << dirEntry->d_name << endl;
31      }
32
33      closedir(dirp);
34      return 0;
35  }
36
```

1.31 Directory I/O detail

Directory I/O detail

```
// open directory
dirp = opendir(argv[1]);
if (dirp == 0) {
   perror(argv[1]);
   exit(EXIT_FAILURE);
}
while ((dirEntry = readdir(dirp)) != NULL) {
   cout << dirEntry->d_name << endl;
}
closedir(dirp);
```

1.32 readDir.cxx

```
readDir.cxx
/*
 * readDir.cxx
 *
 * sample program to list directory content
 *
 */
#include <dirent.h>
#include <cstdio>
#include <cstdlib>
#include <iostream>
using namespace std;

int main(int argc, char* argv[]) {
    if (argc != 2) {
        cerr << "USAGE: readDir pathname\n";
        exit(EXIT_FAILURE);
    }

    DIR *dirp;
    struct dirent *dirEntry;

    // open directory
    dirp = opendir(argv[1]);
    if (dirp == 0) {
        perror(argv[1]);
        exit(EXIT_FAILURE);
    }

    while ((dirEntry = readdir(dirp)) != NULL) {
        cout << dirEntry->d_name << endl;
    }

    closedir(dirp);
    return 0;
}

line: 1 / 36   col: 0   sel: 0   INS   TAB   mode: CRLF   encoding: UTF-8   filetype: C++   scope: unknown
```

1.33 Summary

Summary

- C++ programming with C Library functions
- Next:
  - C++ programming with System Calls