Programming and Algorithms

COMP1038.PGA

Week 11 – Lecture 2: Few advanced concepts

Dr. Pushpendu Kar

Compiling Multiple-Source-File Programs

- You can build a program that consists of multiple source files.
- But the definition of a function must be entirely contained in one file.
- Variable defined outside any function definition are referred to as *Global Variable*.
- Global variables are accessible to any function defined in the same file after the variable is declared.
- However, the global variables must be declared in each file in which they are used.

```
extern int flag;
```

- 'extern' is an storage class specifier
- This indicates that the variable *flag* is defined either later in the same file or in a different file.

Compiling Multiple-Source-File Programs cont.

- Function prototype can extend the scope of a function beyond the file in which it's define.
- You need to include function prototype in the file in which the function is invoked.
- Include the file holding function definition in the file where the function is invoked by the following statement.

```
#include "filename"
```

- Compile the files together.

Restricting Scope with static

- *static* is a storage class specifier.
- When *static* applies to a global variable or a function, prevent it from being used by any function that's not defined in the same file.

static const double PI = 3.14159

- PI is known only to functions in the file in which it is defined

Program termination with exit() and atexit()

- exit() causes a program to terminate immediately.
- This function takes as argument an integer parameter or a symbolic constant like EXIT_SUCCESS or EXIT_FAILURE.
- *atexit()* registers a function that should be called when the program terminated by reaching the end of main or when exit is invoked.
- This function takes as an argument a pointer to a function (i.e., the function name)
- Any function previously registered with *atexit()* are invoked in the reverse order of their registration.
- The function called by *atexit()* cannot have any argument and return value.
- 'stdlib.h' header file provides both the functions.

Program termination with exit() and atexit() cont…

```
#include<stdio.h>
#include<stdlib.h>

void print(void); //prototype

int main(void)
{
   atexit(print); //register function print
   puts(" Enter 1 to terminate program with function exit\nEnter 2 to terminate program normally");
   int answer;
   scanf("%d", &answer);

      If (answer == 1){
        puts("\n Terminating program with function exit");
         exit(EXIT_SUCCESS);
      }

  puts("\nTerminating program by reaching the end of main");
}

void print (void)
{
  puts("Executing function print at program termination\nProgram terminated");
}
```

Signal Handling

- An external asynchronous event, or signal, can cause a program to terminate prematurely.
- Some events include interrupts
  - <Ctrl> c on a Linux/Unix or Windows system
  - <Command> c OS X
- <signal.h> provides the capability to trap unexpected events with function *signal*.
- Function *signal* receives two arguments
  - An integer signal number
  - A pointer to the signal handling function
- Function *signal* should be the first statement in the main function.
- Signal can be generated by function *raise*.

Signal Handling cont…

- Standard signals:

Signal	Explanation
SIGABRT	Abnormal termination of the program (such as a call to function abort).
SIGFPE	An erroneous arithmetic operation, such as divide-by-zero or and operation resulting in overflow.
SIGILL	Detection of illegal instruction
SIGINT	Receipt of an interactive attention signal (<Ctrl> c or <Command> c)
SIGSEGV	An attempt to access memory that is not allocated to a program
SIGTERM	A termination request sent to the program.

Signal Handling cont…

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>

void sighandler(int);

int main () {
  signal(SIGINT, sighandler);

  while(1) {
    printf("Going to sleep for a second...\n");
    sleep(1);
  }
  return(0);
}

void sighandler(int signum) {
  printf("Caught signal %d, coming out...\n", signum);
  exit(1);
}
```

Output:

```
Going to sleep for a second...
Going to sleep for a second...
Going to sleep for a second...
Going to sleep for a second...
Going to sleep for a second...    Press <Ctrl> c
Caught signal 2, coming out...
```

Thank you