CS 354 - Machine Organization & Programming
Tuesday, December 3, 2019

Project p6 (4.5%): DUE at 10 pm on Saturday, December 14th

Homework hw8 (1.5%): Due at 10 pm on Friday, December 6th

Last Time

Meet Signals
Three Phases of Signaling
Processes IDs and Groups
Sending Signals
Receiving Signals

Today

Issues with Multiple Signals

Forward Declaration
Multifile Coding
Multifile Compilation
Makefiles

Next Time

Bring your devices to fill out online course evaluation
Linking and Symbols
Read: B&O 7.3 - 7.6

Issues with Multiple Signals

What? Multiple signals of the same type as well as those of different types

Some Issues

→ Can a signal handler be interrupted by other signals?

❋ *Block any signals*

→ Can a system call be interrupted by a signal?

slow system calls

→ Does the system queue multiple standard signals of the same type for a process?

❋ *Your signal handler shouldn't assume*

Real-time Signals

-
- Multiple signals of same type
- Multiple signals of different types

Forward Declaration

What? Forward declaration

❋ *C requires that an identifier*

Why?

-
-
-

Declaration vs. Definition

declaring

variables:

functions:

defining

variables:

functions:

❋ *Variable declarations*

```
void f(){
   int i = 11;
   static int j;
```

❋ *A variable is proceeded with*

Multifile Coding

What? Multifile coding

Header File (finename.h) - “public” interface

recall **heapAlloc.h** from project p3:

```
#ifndef __heapAlloc_h__
#define __heapAlloc_h__

int   initHeap(int sizeOfRegion);
void* allocHeap(int size);
int   freeHeap(void *ptr);
void  dumpMem();

#endif // __heapAlloc_h__
```

- *An identifier*

  #include guard:

Source File (filename.c) - ”private” implementation

recall **heapAlloc.c** from project p3:

```
#include <unistd.h>
. . .
#include "heapAlloc.h"

typedef struct blockHeader {
   int size_status;
} blockHeader;

blockHeader *heapStart = NULL;

void* allocHeap(int size) { . . . }
int   freeHeap(void *ptr) { . . . }
int   initHeap(int sizeOfRegion) { . . . }
void  dumpMem() {. . . }
```

Multifile Compilation

gcc Compiler Driver

preprocessor

compiler

assembler

linker

Object Files

relocatable object file (ROF)

executable object file (EOF)

shared object file (SOF)

Compiling All at Once

```
gcc align.c heapAlloc.c  -o align
```

Compiling Separately

```
gcc -c align.c
gcc -c heapAlloc.c
gcc align.o heapAlloc.o -o align
```

- *Compiling separately is*

Makefiles

What? Makefiles are

-
-

Why?

-
-

Rules

Example

```
#simplified p3 Makefile
align: align.o heapAlloc.o
     gcc align.o heapAlloc.o -o align
align.o: align.c
     gcc -c align.c
heapAlloc.o: heapAlloc.c heapAlloc.h
     gcc -c heapAlloc.c
clean:
     rm *.o
     rm align
```

Using

```
$ls
align.c  Makefile  heapAlloc.c  heapAlloc.h
$make
gcc -c align.c
gcc -c heapAlloc.c
gcc align.o heapAlloc.o -o align
$ls
align  align.c  align.o  Makefile  heapAlloc.c  heapAlloc.h  heapAlloc.o
$rm heapAlloc.o
rm: remove regular file 'heapAlloc.o'? y
$make
gcc -c heapAlloc.c
gcc align.o heapAlloc.o -o align
$make heapAlloc.o
make: 'heapAlloc.o' is up to date.
$make clean
rm *.o
rm align
$ls
align.c  Makefile  heapAlloc.c  heapAlloc.h
```