LECTURE 6: C PROGRAMMING

Computer Systems and Networks

Dr. Pallipuram
(vpallipuramkrishnamani@pacific.edu)

University of the Pacific

Today’s Class

- More String Operations
- Insight into malloc and free
- file I/O

ctype Library

Useful for character manipulation

```
#include <ctype.h>
```

`toupper(char)` / **`tolower(char)`** – Converts character to uppercase or lowercase

- Example:

```
char c = toupper('a');
printf("%c", c);  // A
```

ctype Library

`isalpha(char)` – Is the character a letter?

`isdigit(char)` – Is the character a number 0-9?

`isspace(char)` – Is the character whitespace? (space or newline character)

`ispunct(char)` – Is the character punctuation? (technically, a visible character that is not whitespace, a letter, or a number)

… and several other variations

What is the output of the C snippet below?

```
main( )
{
char s[ ] = "CPP programmers!
You know C!" ;
printf ( "\n%s", &s[2] ) ;
printf ( "\n%s", s ) ;
printf ( "\n%c", s[2] ) ;
}
```

What’s the Error?

```
char *a = malloc(128*sizeof(char));
char *b = malloc(128*sizeof(char));
b = a;
free(a);
free(b);
```

http://www.yolinux.com/TUTORIALS/C++MemoryCorruptionAndMemoryLeaks.html

What’s the (Potential) Error?

```
char *a = malloc(128*sizeof(char));

dataLen = <some value...>

// Copy “dataLen” bytes
// starting at *data to *a
memcpy(a, data, dataLen);
```

http://www.yolinux.com/TUTORIALS/C++MemoryCorruptionAndMemoryLeaks.html

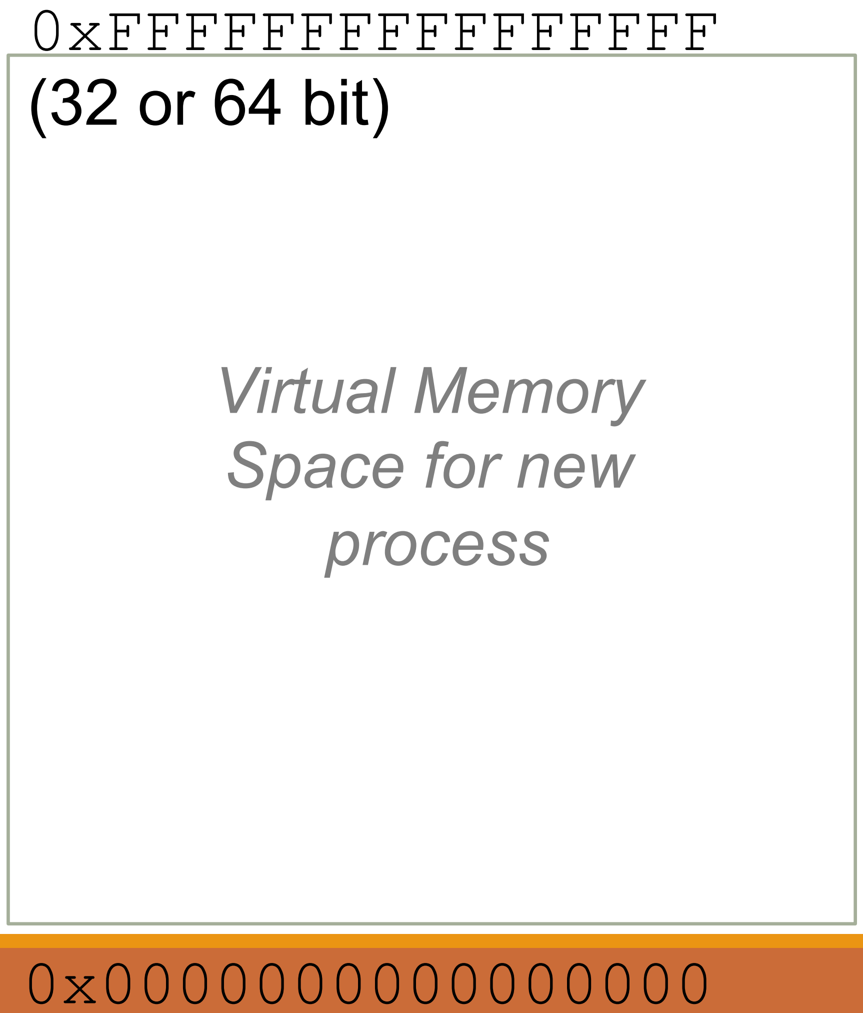
C program memory management

Memory Management

OS creates **virtual memory** space for process when started

Region is huge (full 32 or 64 bit space)

- **Not** fully mapped to physical memory
- Otherwise you could only fit 1 program in memory



Memory Management

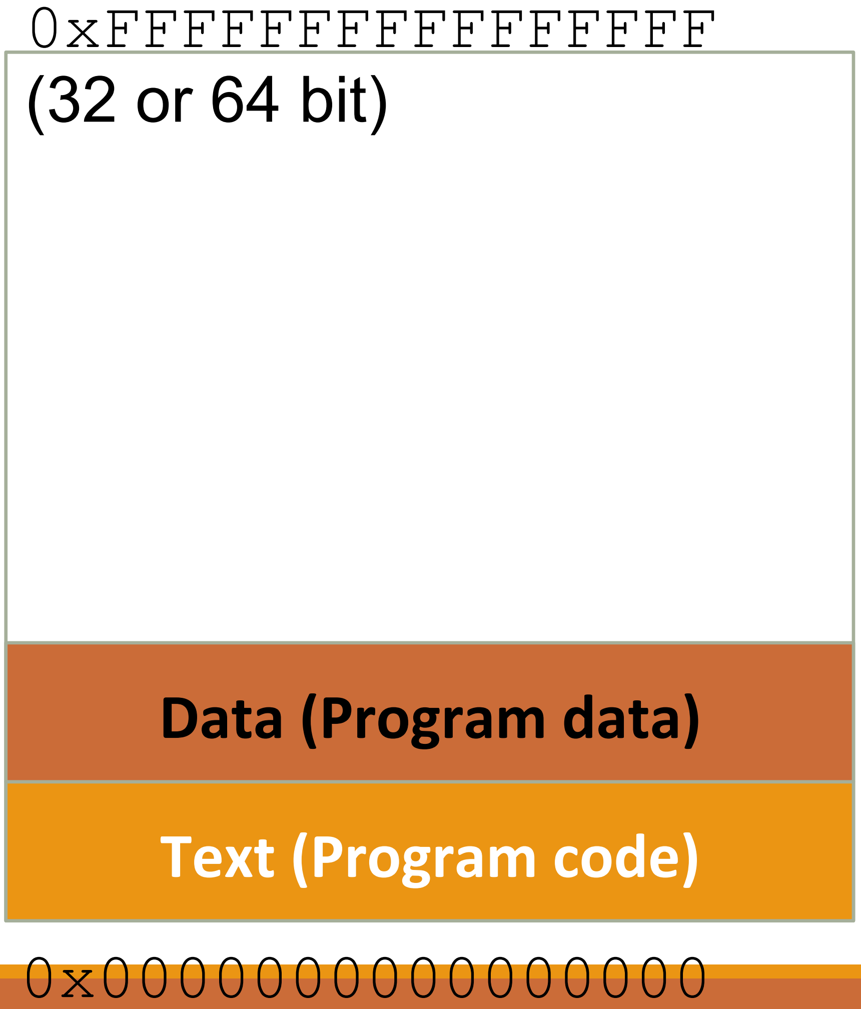
OS loads in the program from disk

"Text" region

- Program **code**

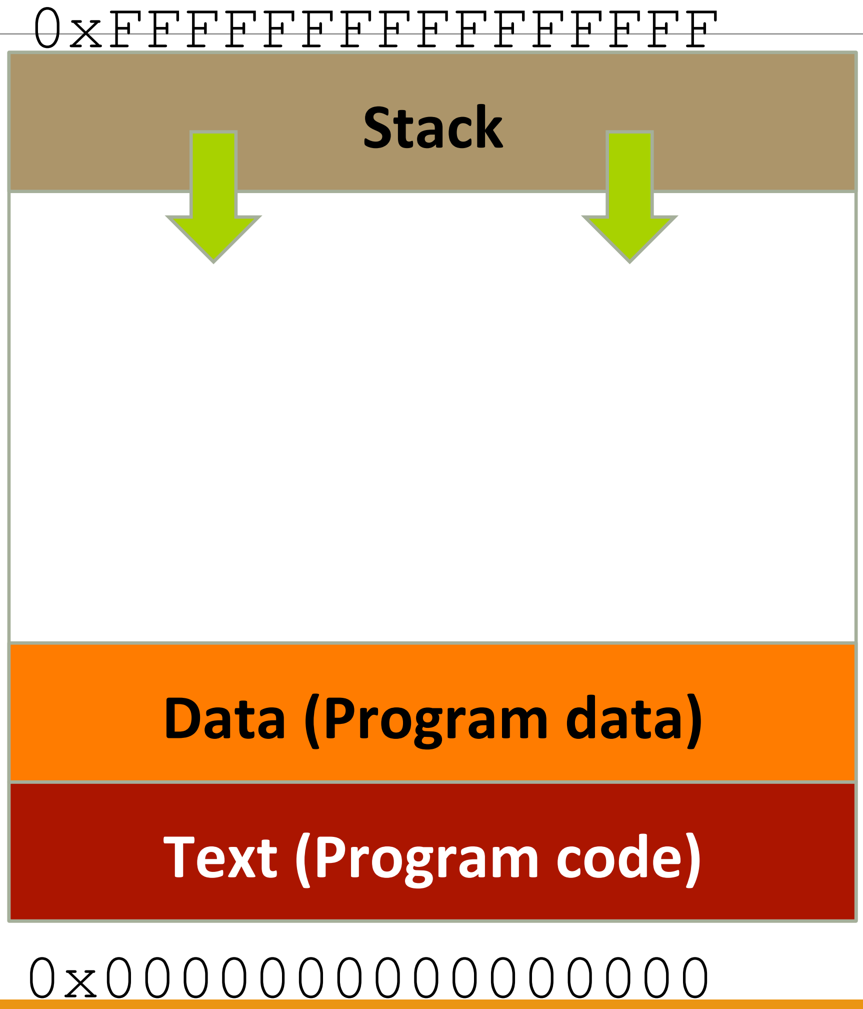
"Data" region

- Program fixed **data**



Memory Management

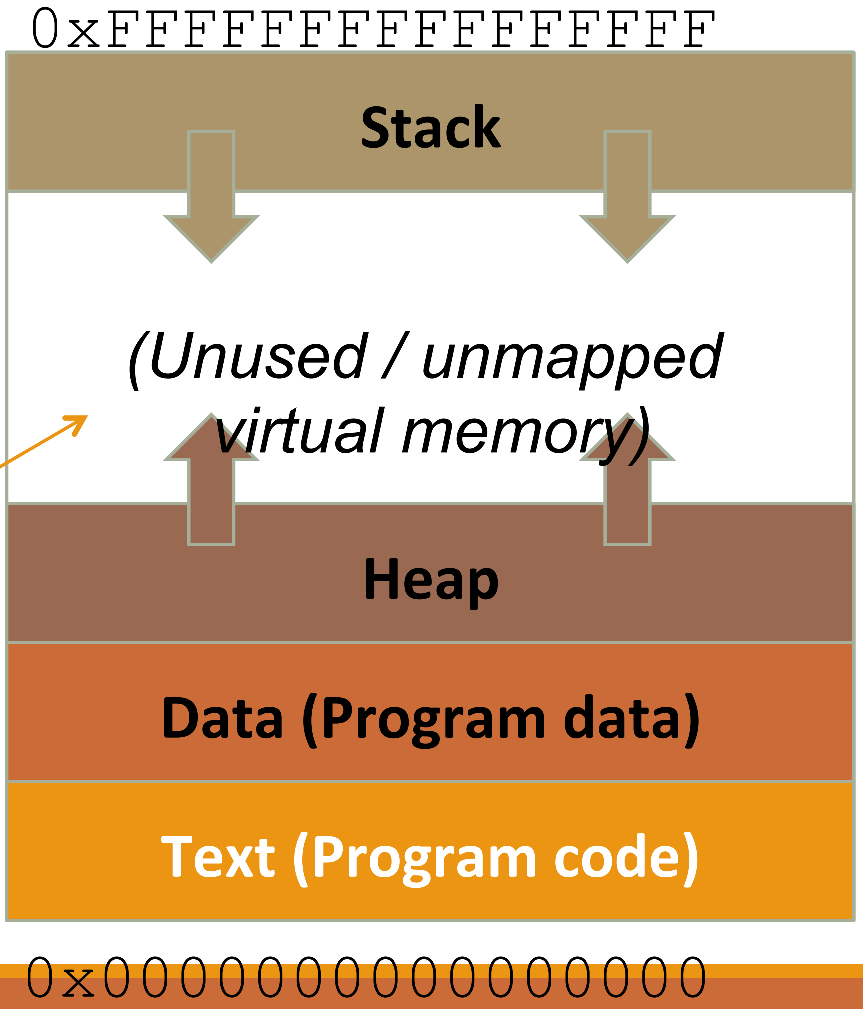
Stack created to track program function calls and local variables



Memory Management

Heap created to store dynamic memory from `malloc()` and related functions

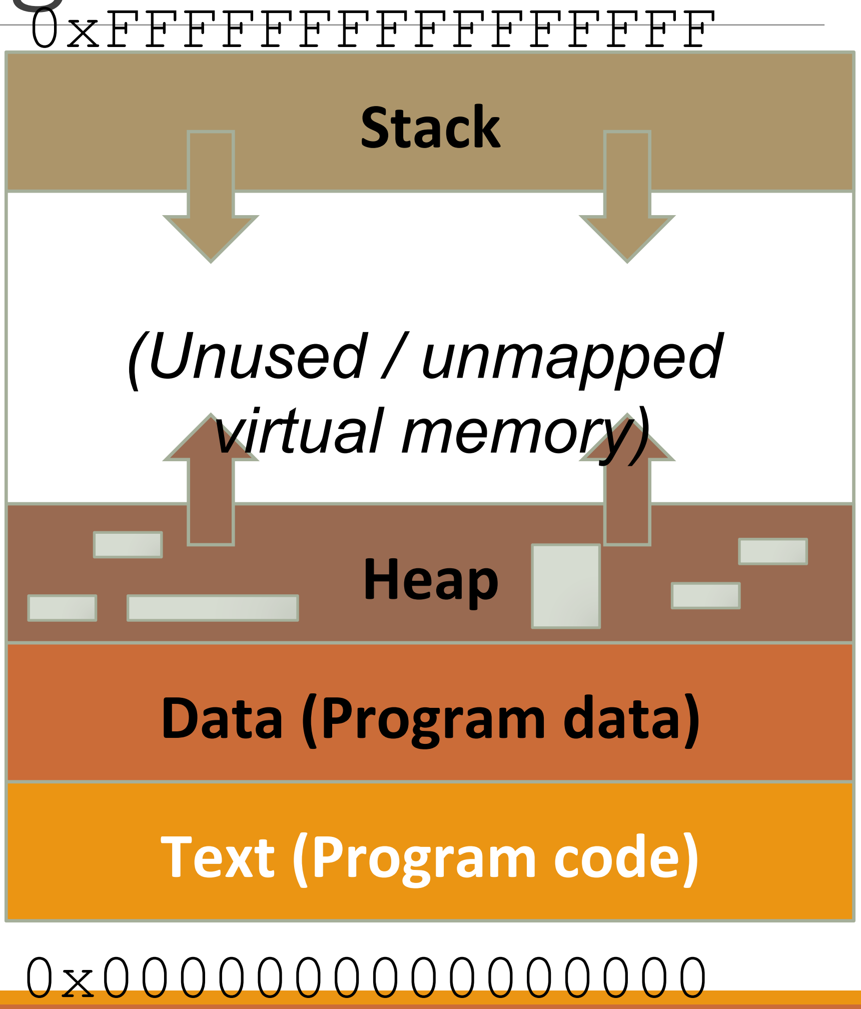
Not to scale – this unused region is **huge!**



Memory Management

Program starts running

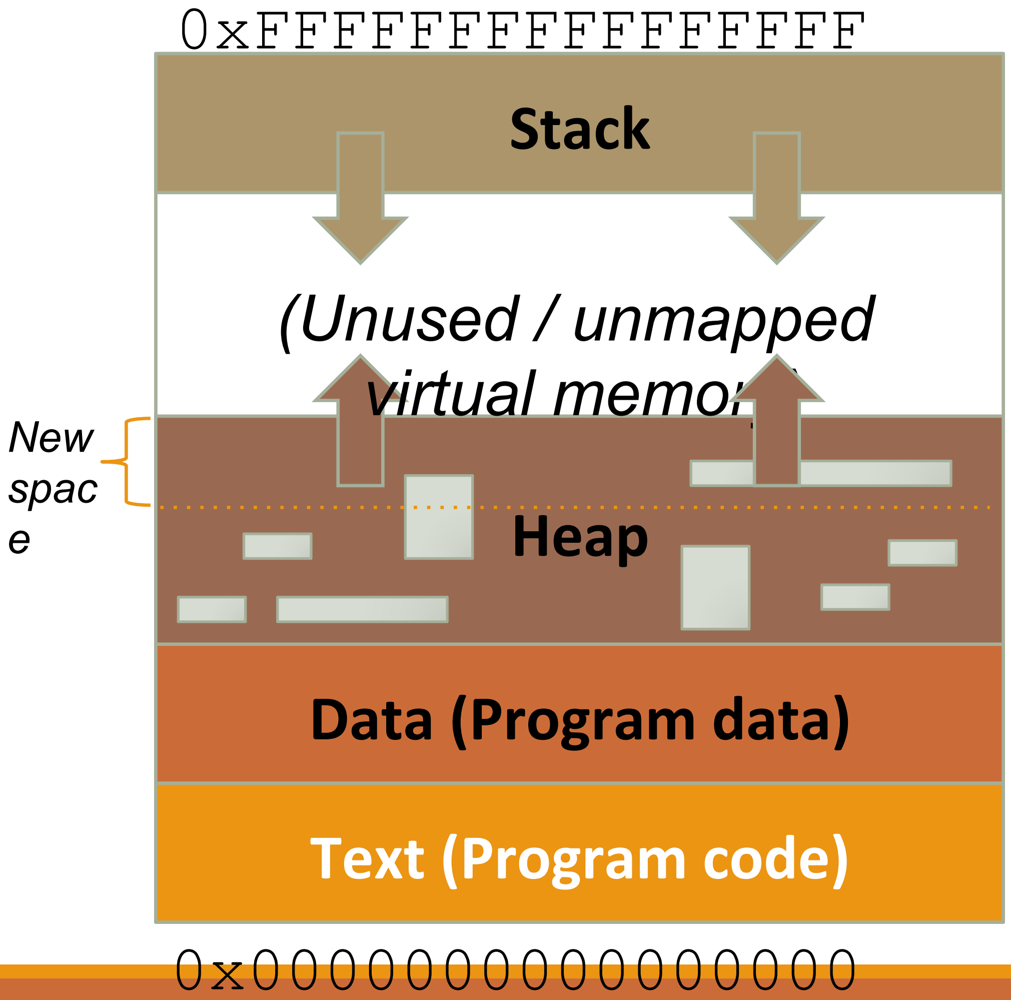
`malloc()` allocates some memory



Memory Management

Original heap space eventually fills up

`malloc()` *requests* additional space from the kernel by using `brk()` system call

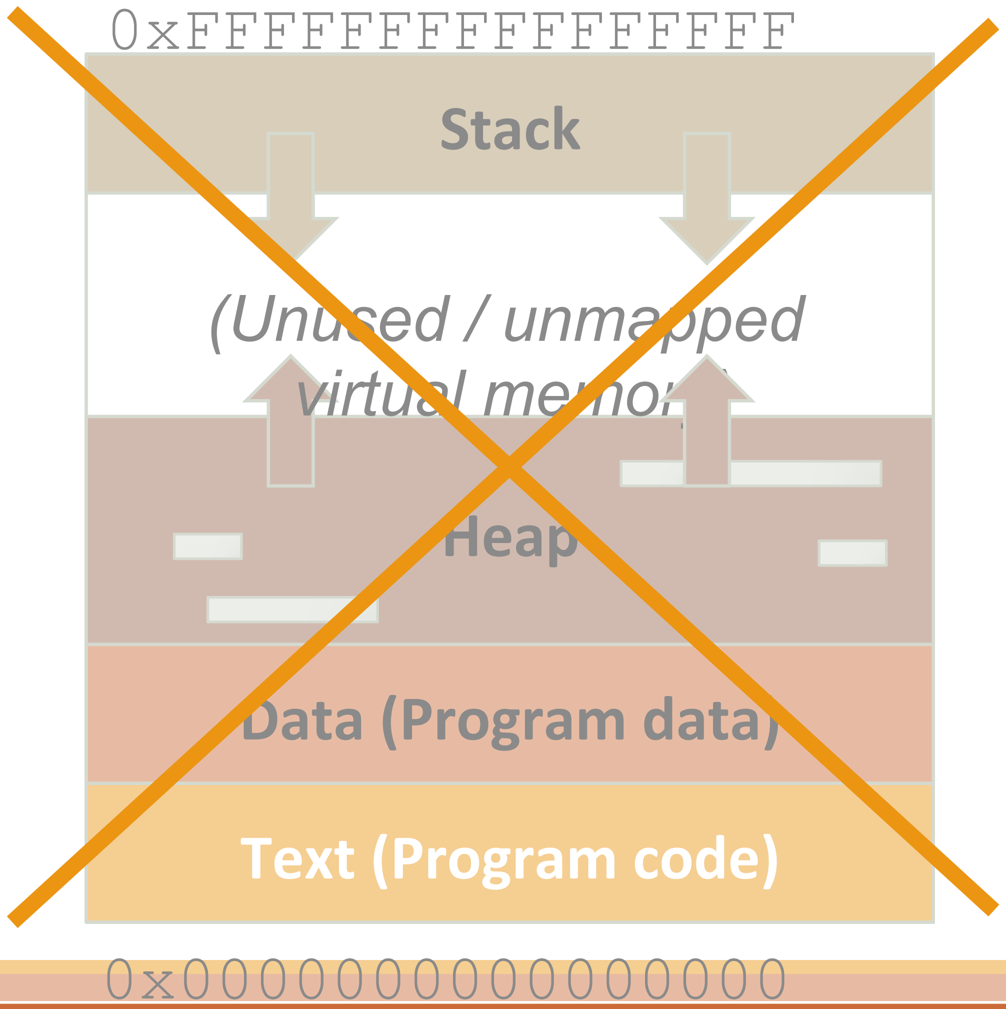


Memory Management

Program terminates

OS expunges entire virtual address space

- Everything is deleted



File I/O

LET US GO OVER AN EXAMPLE

```
#include<stdio.h>

     int main()
     {
          FILE *ptr_file;
          char buf[1000];

          ptr_file =fopen("input.txt","r");
          if (!ptr_file)
               return 1;

          while (fgets(buf,1000, ptr_file)!=NULL)
               printf("%s",buf);

          fclose(ptr_file);
          return 0;
     }
```

File I/O Functions

- fopen – opens a text file.
- fclose – closes a text file.
- feof – Google it!
- fgets – reads a string from a file.
- fwrite – Google it!
- fgetc – reads a character from a file.
- fputc – prints a character to a file.

You are ready for Lab 4!