Hw0

Lanxiao Bai NetID:lbai5

Chapter 1

* ＃include <unistd.h>

int main()

{

write(1, “Hi! My name is Lanxiao Bai.\n”, 28);

return 0;

}

* ＃include <unistd.h>

＃include <stdlib.h>

int main(int argc, char\* argv[])

{

int n = atoi(argv[1]);

if (n <= 0)

return 0;

else

{

for (int i = 1; i <= n; i++)

{

for (int j = 0; j < i; j++)

{

write(STDERR\_FILENO, “\*”, 1);

}

write(STDERR\_FILENO, “\n”, 1);

}

return 0;

}

}

* #include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

int main()

{

mode\_t mode = S\_IRUSR | S\_IWUSR;

int fileCreated = open("output.txt", O\_RDWR | O\_CREAT | O\_TRUNC, mode);

write(fileCreated, "Hello World.\n", 12);

close(fileCreated);

return 0;

}

* ＃include <stdio.h>

int main()

{

printf(“Hello World.\n”);

return 0;

}

* printf() is a wrapped function of write()
* printf() is called when buffer is full

Chapter 2

* + 8
  + 1
  + Int: 4, Double: 8, Float: 4, Long: 8, Long Long: 8
  + 0x7fbd9d50
  + data+3
* + “hello” is stored on heap and was never freed
  + 5
  + 13
  + “abc”
  + 1

Chapter 3

* + - Traverse argv
    - Atoi()
  + Excutable File’s name
* char\*\* environ
* 8 – the size of pointer, 6 – 5 chars and 1 \0
* stack

Chapter 4

* + In the scope of function, declare it as static variable
  + free
* + There’s no enough memory to allocate, time() returns the number of second since 1970

ctime() returns the time as a string

* + time() returns the number of second since 1970

ctime() returns the time as a string

* + It freed the same part of memory twice
  + There’s no enough memory to allocate
  + Use memory that’s already been freed
  + Match free() with malloc()
* + struct Person{

char\* name;

unsigned int age;

Person\* friends;

int count = 0;

};

typedef Person person\_t;

* person\_t\* ptr1 = (person\_t\*)malloc(sizeof(person\_t));

person\_t\* ptr2 = (person\_t\*)malloc(sizeof(person\_t));

ptr1->name = “Agent Smith”

ptr2->name = "Sonny Moore"

ptr1->age = 128;

ptr2->age = 256;

ptr1->friends = (person\_t\*)malloc(10 \* sizeof(person\_t));

ptr2->friends = (person\_t\*)malloc(10 \* sizeof(person\_t));

ptr1->friends[count++] = ptr2;

ptr2->friends[count++] = ptr1;

* + person\_t\* creat(char \*name, int age, person\_t friend)

{

person\_t\* ptr1 = (person\_t\*)malloc(sizeof(person\_t));

ptr1->name = strdup(name);

ptr1->age = age;

ptr1->friends = (person\_t\*)malloc(10 \* sizeof(person\_t));

ptr1->friend[count++] = friend;

return ptr1;

}

* void destroy(person\_t\* person)

{

if (person == NULL)

return;

free(name);

for (int i = 0; i < 10; i++)

{

free(person->friends[i]);

person->friends[i] = NULL;

}

free(person);

}

Chapter 5

* + gets(), puts()
  + could cause buffer overflow
* char \*data = “Hello 5 World”;

char a[6];

int b;

char c[6];

sscanf(data, “%s %d %s”, a, b, c);

* + GNU\_SOURCE\_
  + #define \_GNU\_SOURCE

#include <stdio.h>

#include <stdlib.h>

int main()

{

FILE\* file = fopen(“input.txt”, “r”);

char\* buffer = NULL;

getline(& buffer, file);

printf(“%s\n”, buffer);

fclose(file);

return EXIT\_SUCCESS;

}

C Develop

* -g
* Because the code itself wasn’t modified
* Tabs
* Stack memory is automatically handled by program, while heap memory needed to be allocated and freed manually.
* Text Segment, Data Segment, BSS Segment