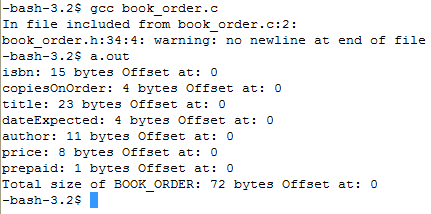
Od Out: Jan 20, 2015

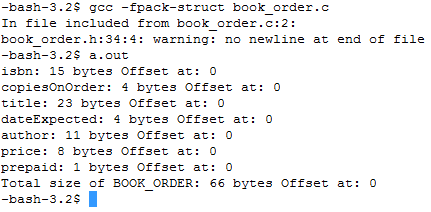
Due: Feb 10, 2015

**CENG251 – Lab #2**

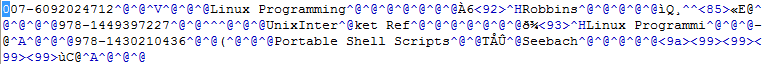
**Part I: Investigating the memory layout of a struct (19 marks)**

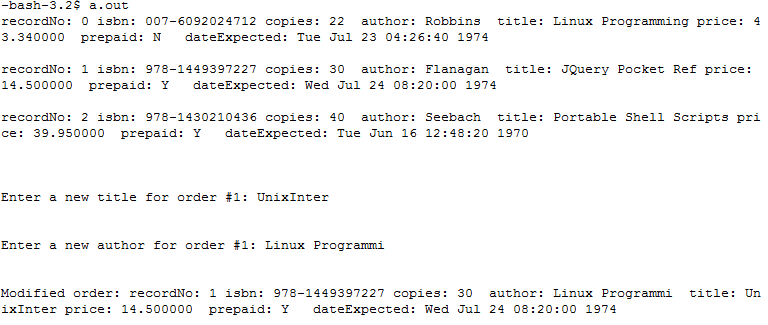
1. Review the 4 allTypes programs. Write a short program using the file **book\_order.h** that determines the following: (8 marks)



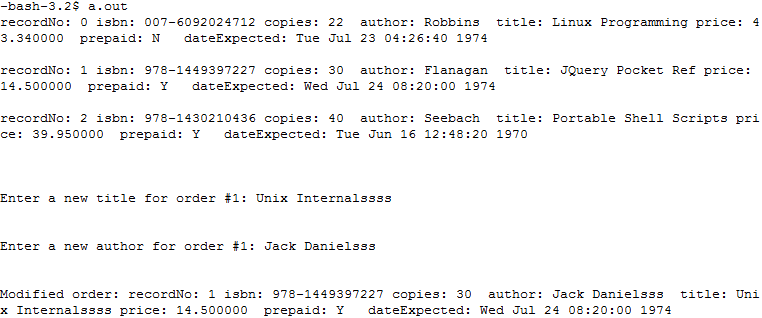
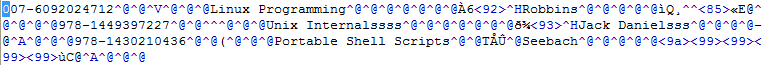


1. We are going to illustrate a buffer overrun in C. This buffer overrun will not do anything dangerous other than unintentionally overwrite some data.   
     
   Run the program book\_order.c and enter a new author (no more than 10 characters) and title of the 1st record with a title that is as least 2 characters shorter than what is currently present. Examine the dump file and verify the changes.





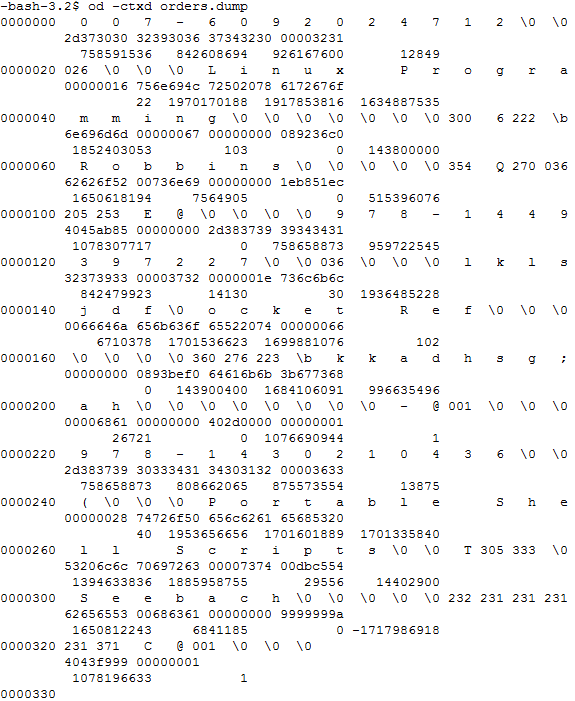
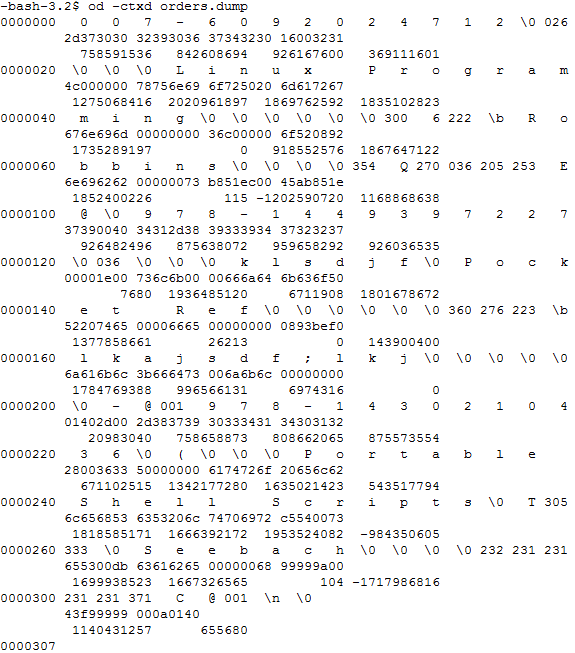
Now run the program again with an title longer than 15 characters and an author is also longer than 11 characters. Describe the problem.



What’s the difference if between running the program with or without the **–fpack-struct** option? (2 marks)

1. Use **od -ctxd** to examine the dumpfile from the program when using and not using –fpackstruct. In each case how big is the file? In each case at which byte offsets does each flight record begin at? In each case which bytes get overwritten by the overly long input. (3 marks)

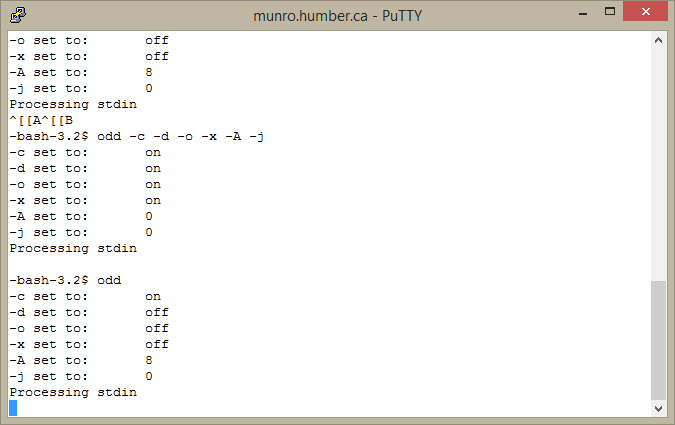
The file is 27 bytes bigger than when I use –fpack-struct

  
without –fpack-struct with –fpack-struct

1. You’ll also observe that compiling the program with **gets** generates a warning. Look up how to use **fgets** and replace the gets function – were you able to solve the problem? (Provide a listing and a write up of observations: 2 marks)

Yes I was all I had to do was give how many arguments will be taken in and write stdin at the end  
**Part II: Writing Octal Dump in C (17 marks)**

1. Write a program that captures and verifies the following flags as they are used in the program **od: -c -d -o -x -A -j** (6 marks)



Code

#include <stdio.h>

#include "book\_order.h"

#define NEWLINE "\n"

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

{

int offset = 0;

struct BOOK\_ORDER order;

struct BOOK\_ORDER orders[]={

{"007-6092024712",22,"Linux Programming", 143800000,"Robbins",43.34,0},

{"978-1449397227",30,"JQuery Pocket Ref", 143900400,"Flanagan",14.50,'Y'},

{"978-1430210436",40,"Portable Shell Scripts",14402900,"Seebach",39.95,'Y'}} ;

fprintf(stdout,"isbn: %d bytes Offset at: %d\n",sizeof(order.isbn), offset);

fprintf(stdout,"copiesOnOrder: %d bytes Offset at: %d\n",sizeof(order.copiesOnOrder), offset);

fprintf(stdout,"title: %d bytes Offset at: %d\n",sizeof(order.title), offset);

fprintf(stdout,"dateExpected: %d bytes Offset at: %d\n",sizeof(order.dateExpected), offset);

fprintf(stdout,"author: %d bytes Offset at: %d\n",sizeof(order.author), offset);

fprintf(stdout,"price: %d bytes Offset at: %d\n",sizeof(order.price), offset);

fprintf(stdout,"prepaid: %d bytes Offset at: %d\n",sizeof(order.prepaid), offset);

fprintf(stdout,"Total size of BOOK\_ORDER: %d bytes Offset at: %d\n",sizeof(order), offset);

return 0;

}

Odd.c

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Harsh Joshi

Tuesday February 10, 2015

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <errno.h>

#include <stdio.h>

#include <getopt.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include <limits.h>

struct GetOptFlags //structure is defined globally

{

bool asciiFormat;

bool decimalFormat;

bool octalFormat;

bool hexFormat;

int addressRadix;

int skipBytes;

char \* fileName;

FILE \* outputDevice;

};

void do\_Od(FILE \*fp,struct GetOptFlags flags)

{

char buffer[512]="";

int lineCount=0;

int counter2=0;

(fseek(fp,flags.skipBytes,SEEK\_SET));//skips specified number of bytes

//Flags work together. Process all the flags in the same loop

while(fgets(buffer,sizeof(buffer),fp))

{

fprintf(flags.outputDevice,"%s\n",buffer);

}

fprintf(flags.outputDevice,"\n\n\n"); //Add spacing after the end of the file

}

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

{

//A bundle of variables to remember the flags on the

//command line

struct GetOptFlags getOptFlags =

{true,false,false,false, 8, 0,NULL,stdout};

unsigned int flag,indexVal;

char \* shortOptions="cdoxA:j:"; //Note - none of these has parameters

//process options

while( (flag=getopt(argc,argv,shortOptions))!=-1)

{

switch(flag)

{

case 'c': getOptFlags.asciiFormat=true;

break;

case 'd': getOptFlags.decimalFormat=true;

break;

case 'o': getOptFlags.octalFormat=true;

break;

case 'x': getOptFlags.hexFormat=true;

break;

case 'A': getOptFlags.addressRadix=atoi(optarg);

break;

case 'j': getOptFlags.skipBytes=atoi(optarg);

break;

}

}

//Verify the options

fprintf(stdout, "-c set to:\t %s\n", getOptFlags.asciiFormat ? "on" : "off");

fprintf(stdout, "-d set to:\t %s\n", getOptFlags.decimalFormat ? "on" : "off");

fprintf(stdout, "-o set to:\t %s\n", getOptFlags.octalFormat? "on" : "off");

fprintf(stdout, "-x set to:\t %s\n", getOptFlags.hexFormat ? "on" : "off");

fprintf(stdout, "-A set to:\t %d\n", getOptFlags.addressRadix);

fprintf(stdout, "-j set to:\t %d\n", getOptFlags.skipBytes);

if(optind==argc) {

fprintf(getOptFlags.outputDevice,"Processing stdin\n");

do\_Od(stdin,getOptFlags); //No args - read from terminal

}

else while(argv[optind]) //Process each of the files

{

FILE \*infile;

fprintf(getOptFlags.outputDevice,"Processing file: %s\n\n",argv[optind]);

infile=fopen(argv[optind],"r");

if(infile) {

do\_Od(infile,getOptFlags);

fclose(infile);

}

else {

fprintf(stderr,"Error code: %d ",errno);

perror("Oops ");

}

optind++;

}

return 0;

}

Modify book order

#include <stdio.h>

#include "book\_order.h"

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

{

struct BOOK\_ORDER order;

struct BOOK\_ORDER orders[]={

{"007-6092024712",22,"Linux Programming", 143800000,"Robbins",43.34,0},

{"978-1449397227",30,"JQuery Pocket Ref", 143900400,"Flanagan",14.50,'Y'},

{"978-1430210436",40,"Portable Shell Scripts",14402900,"Seebach",39.95,'Y'}

} ;

//Show all the records

displayOrder(0,orders[0]);

displayOrder(1,orders[1]);

displayOrder(2,orders[2]);

//Prompt for a replacement title and author

fprintf(stdout,"\n\nEnter a new title for order #1: " );

fflush(stdout);

fgets(orders[1].title, 11, stdin);

fprintf(stdout,"\n\nEnter a new author for order #1: " );

fflush(stdout);

fgets(orders[1].author, 8, stdin);

fprintf(stdout,"\n\nModified order: ");

displayOrder(1,orders[1]);

FILE \* dumpFile=fopen("orders.dump","wb");

fwrite(orders,sizeof order,sizeof orders/sizeof order,dumpFile);

fclose(dumpFile);

return 0;

}