Structs

EECS 2031

Song Wang
wangsong@eecs.yorku.ca
eecs.yorku.ca/~wangsong/

Acknowledgement

- Some of the covered materials are based on previous EECS2031 offerings:
  - Uyen Trang (UT) Nguyen, Pooja Vashisth, Hui Wang, Manos Papagelis

debug with gdb

```c
# include <stdio.h>

int main()
{
    int i, num, j;
    printf ("Enter the number: ");
    scanf ("%d", &num );

    for (i=1; i<num; i++)
        j=j*i;

    printf("The factorial of %d is %d\n",num,j);
}
```

```
Enter the number: 3
The factorial of 3 is 0
```

https://u.osu.edu/cstutorials/2018/09/28/how-to-debug-c-program-using-gdb-in-6-simple-steps/

Launch gdb

```
indigo 309 % gcc -g test.c
indigo 310 % gdb a.out
GNU gdb (GDB) Red Hat Enterprise Linux 8.2-19.el8
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...done.
(gdb)
```

Set up a break point inside C program

Syntax:

```
break line_number
```

```
(gdb) break 10
Breakpoint 1 at 0x40062c: file test.c, line 10.
(gdb)
```

Execute the C program in gdb debugger

```
run [args]
```

```
(gdb) run 4
Starting program: /eecs/home/wangsong/eecs2031/a.out 4
Enter the number: 4

Breakpoint 1, main () at test.c:10
10                      j=j*i;
Missing separate debuginfos, use: yum debuginfo-install glibc-2.28-225.el8.x86_6
4
(gdb)
```

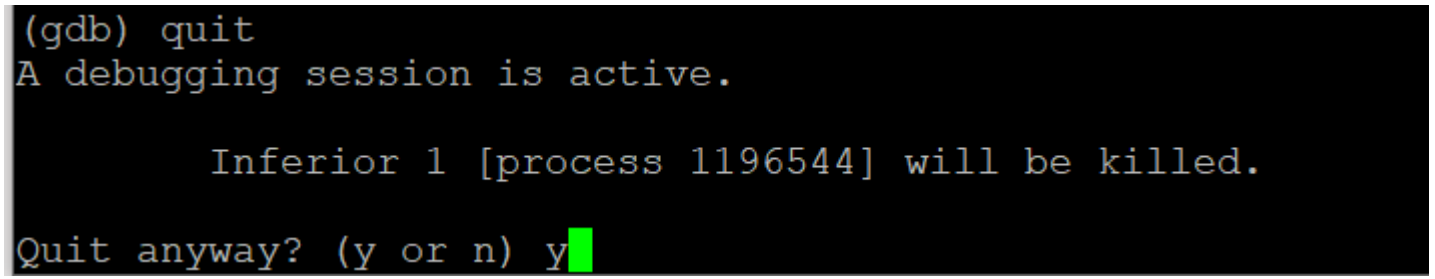
Printing the variable values inside gdb debugger

```
Syntax: print {variable}


Examples:
print i
print j
print num
```

```
Breakpoint 1, main () at test.c:10
10                      j=j*i;
Missing separate debuginfos, use: yum debuginfo-install glibc-2.28-225.el8.x86_6
4
(gdb) print i
$1 = 1
(gdb) print j
$2 = 0
(gdb) print num
$3 = 4
(gdb)
```

Quit the gdb


(gdb) quit
A debugging session is active.
Inferior 1 [process 1196544] will be killed.
Quit anyway? (y or n) y

Structures in C

- Be able to use compound data structures in programs
- Be able to pass compound data structures as function arguments, either by value or by reference
- Be able to do simple bit-vector manipulations

```
struct myStructure {
  int myNum;
  char myLetter;
};

int main() {
  struct myStructure s1;
  return 0;
}
```

Structures

- **Compound data:**
- **A date is**
  - **an `int month` and**
  - **an `int day` and**
  - **an `int year`**

```
struct ADate {
   int  month;
   int  day;
   int  year;
};

struct ADate date;

date.month = 2;
date.day = 4;
date.year = 2021;
```

- Unlike Java, C doesn't automatically define functions for initializing and printing ...

Structure Representation & Size

`sizeof(struct …) =`

sum of `sizeof(field)`

```
struct CharCharInt {
   char  c1;
   char  c2;
   int   i;
} foo;

foo.c1 = 'a';
foo.c2 = 'b';
foo.i  = 0xDEADBEEF;
```

Theoretical:

Structure Representation & Size

`sizeof(struct …) =`

sum of `sizeof`(field)

```
struct CharCharInt {
   char  c1;
   char  c2;
   int   i;
} foo;

foo.c1 = 'a';
foo.c2 = 'b';
foo.i  = 0xDEADBEEF;
```

```c
#include<stdint.h>
#include<stdio.h>

struct carModel
{
    uint32_t carNumber;
    uint32_t carPrice;
    uint16_t carMaxSpeed;
    float    carWeight;
};

int main(void)
{
    struct carModel  carBMW = {2021,15000,220,1330};
    printf("Sizeof of struct carModel is %I64u\n",sizeof(struct carModel));

    getchar();

    return 0;
}
```

```
Sizeof of struct carModel is 16
```

Variations in Struct declaration

- Struct variables can be declared inside other structs

```
struct D {
  int e;
  int f;
}
```

```
struct X {
  int      i;
  struct D d;
  int      j;
}
```

- Struct members can be arrays or pointers

```
struct W {
  int i;
  int a[10];
  int j;
}
```

```
struct V {
  int  i;
  int* a;
  int  j;
}
```

Typedef

- Mechanism for creating new type names
  - give an alias name to an existing type
  - *Create new type*

```
typedef  int  xyz;
xyz i = 0;
```

```
typedef struct ADate {
   int month;
   int day;
   int year;
} Date;

Date d = { 2, 4, 2021 };
```

New type name

Typedef

```c
// defining structure
struct str1 {
    int a;
};

// defining new name for str1
typedef struct str1 str1;

// another way of using
typedef with structures
typedef struct str2 {
    int x;
} str2;
```

```c
int main()
{
    // creating structure
variables using new names
    str1 var1 = { 20 };
    str2 var2 = { 314 };

    printf("var1.a = %d\
n", var1.a);
    printf("var2.x = %d",
var2.x);

    return 0;
}
```

Things you can and can't do

- You can

  Use = to assign whole struct variables
- You can
  - Have a struct as a function return type
- You cannot
  - Use == to directly compare struct variables; can compare fields directly
- You cannot
  - Directly scanf or printf structs; can read fields one by one.

Arrays of Structures

Array declaration

```
Date birthdays[100];

bool check_birthday(Date today)
{
  int i;

  for (i = 0; i < 100; i++) {
    if ((today.month == birthdays[i].month)
&&
        (today.day == birthdays[i].day))
      return true;

  return false;
}
```

Array index, then structure field

Structures and arrays

```c
const int MAX_COUNTRY_NAME_LENGTH = 50;

typedef struct CountryTvWatch_struct {
   char countryName[MAX_COUNTRY_NAME_LENGTH];
   int tvMinutes;
} CountryTvWatch;

int main(void) {
   // Source: www.statista.com, 2010
   const int NUM_COUNTRIES = 4;

   CountryTvWatch countryList[NUM_COUNTRIES];
   char countryToFind[MAX_COUNTRY_NAME_LENGTH];
   bool countryFound;
   int i;

   strcpy(countryList[0].countryName, "Brazil");
   countryList[0].tvMinutes = 222;
   strcpy(countryList[1].countryName, "India");
   countryList[1].tvMinutes = 119;
   strcpy(countryList[2].countryName, "U.K.");
   countryList[2].tvMinutes = 242;
   strcpy(countryList[3].countryName, "U.S.A.");
   countryList[3].tvMinutes = 283;
```

```
Enter country name: U.S.A.
People in U.S.A. watch
283 minutes of TV daily.
...
Enter country name: UK
Country not found, try again.
...
Enter country name: U.K.
People in U.K. watch
242 minutes of TV daily.
```

Initializing Structures

- Structures within structures:

```c
#include <stdio.h>
typedef struct  customer {
    char name [20] ;
    char email [25] ;
} Customer;
typedef struct record {
    Customer buyer ;
    char item [20] ;
    int amount ;
} Record;

int main() {

    Record  mysale = { { "Acme Industries", "George Adams"} ,
                    "Zorgle Blaster",  1000} ;

    printf("%s\n", mysale.buyer.name);

    return 0;
}
```

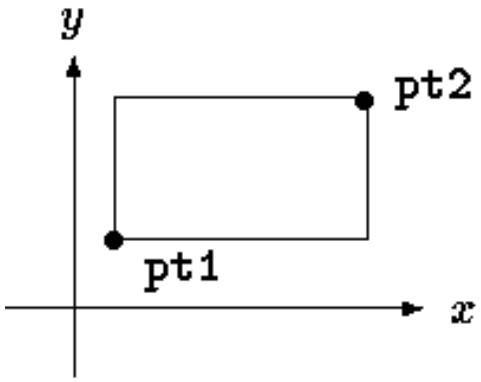
Struct can be used as input/return type

```
struct pairInt {
    int min, max;
};
struct pairInt min_max(int x,int y)
{
    struct pairInt pair;
    pair.min = (x > y) ? y : x;
    pair.max = (x > y) ? x : y;
    return pairInt;
}
int main(){
    struct pairInt result;
    result = min_max( 3, 5 );
    printf("%d<=%d", result.min, result.max);
}
```

Nested Structures

```
struct point {
  int x;
  int y;
};
struct rectangle {
  struct point pt1;
  struct point pt2;
};

struct rectangle screen;
screen.pt1.x = 1;
screen.pt1.y = 2;
screen.pt2.x = 8;
screen.pt2.y = 7;
```

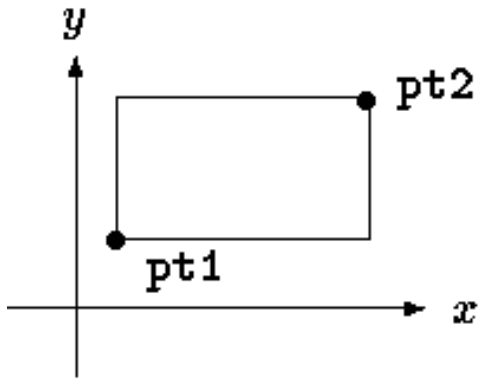


Width? Length? Area?

Nested Structures

```
struct point {
  int x;
  int y;
  int z;
};
struct rectangle {
  struct point pt1;
  struct point pt2;
};

struct rectangle screen;
screen.pt1.x = 1;
screen.pt1.y = 2;
screen.pt2.x = 8;
screen.pt2.y = 7;
```



Width? Length? Area?

Nested Structures

```
struct cube {
  struct rectangle r1;
  struct rectangle r2;
  struct rectangle r3;
  struct rectangle r4;
  struct rectangle r5;
  struct rectangle r6;


};
struct cube c;
c.r1.pt1.x = 1;
c.r1.pt1.y = 2;
c.r1.pt1.z = 0;
...
```