Introduction to Computer Science and Programming in C

Session 9: September 30, 2008

Columbia University

Announcements

- Homework 2 is out. Due 10/14
- Midterm Review on 10/16, exam on 10/21
- Start the homework!
- Submission procedure

Checking your tar file

General Description Requirements Readings Topics Policies Schedule

Introduction to Computer Science and Programming in C

General Information

Instructor: Bert Huang. Office hours Tuesday 2:30 PM - 4:30 CEPSR 624 (or by appointment)
TA: Deergha Sahni, UNI: ds2664, Office hours Wednesday 3:00 PM - 5:00 PM TA Room http://ta.cs.columbia.edu/tamap.shtml
TA: Peter Lu, UNI: yl2505, Office hours Thursday 4:00 PM - 6:00 PM TA Room
TA: Sharath Avadoot Gururaj, UNI: sa2617, Office hours Monday 1:30 PM - 3:30 PM TA Room
Time: Tuesday and Thursday 1:10 PM - 2:25 PM
Location: ~~Mudd 834~~ Changed to **Mudd 233**
Courseworks site (message board etc.): http://courseworks.columbia.edu/

Description

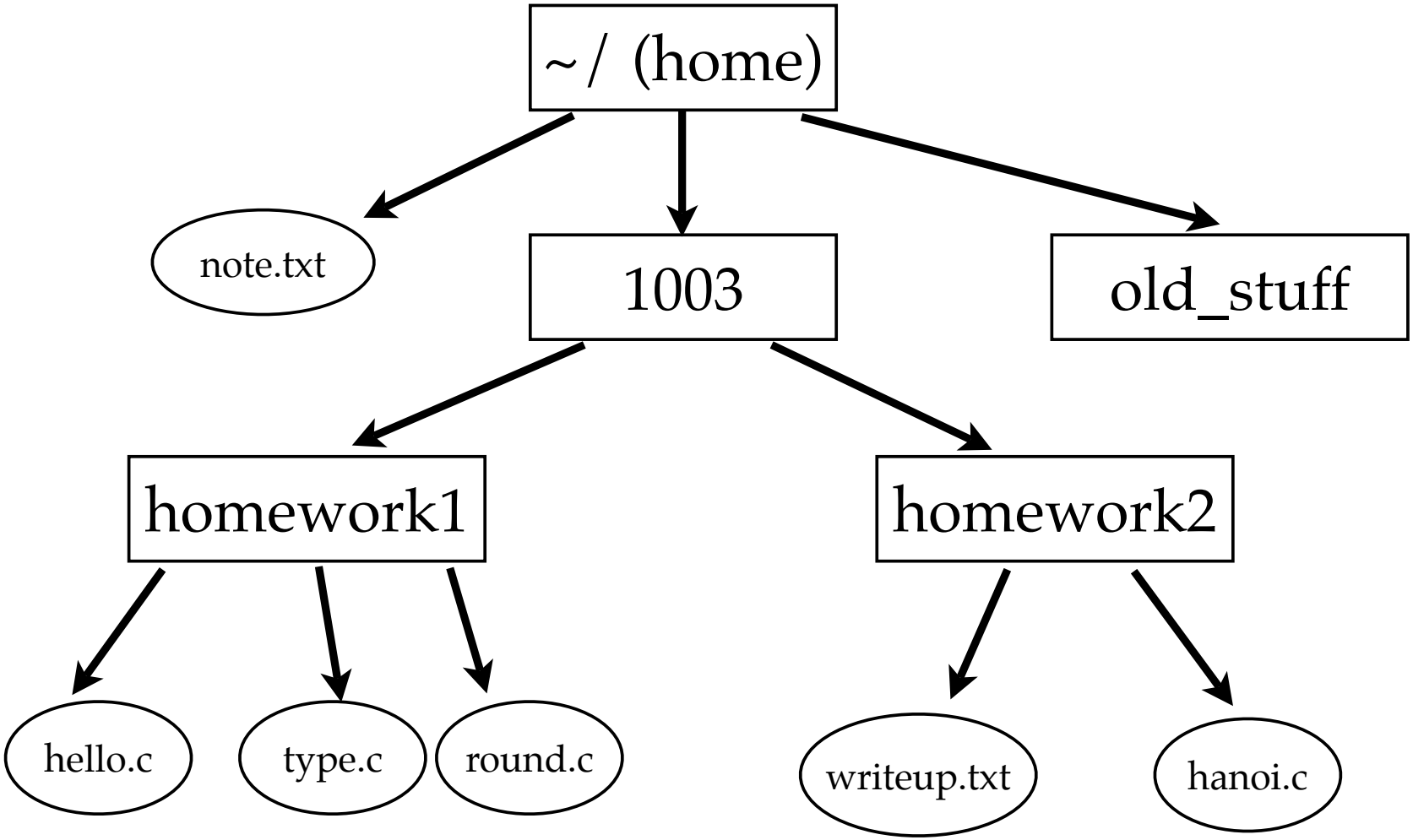
Review

- Finished talking about recursion
- Hanoi solution: Move N discs
  - Move (N-1) discs out of the way
  - Move bottom disc to destination
  - Move (N-1) discs to destination

Today

- Quick useful recursion example
- Advanced types: structs, unions, typedef
- Programming tips

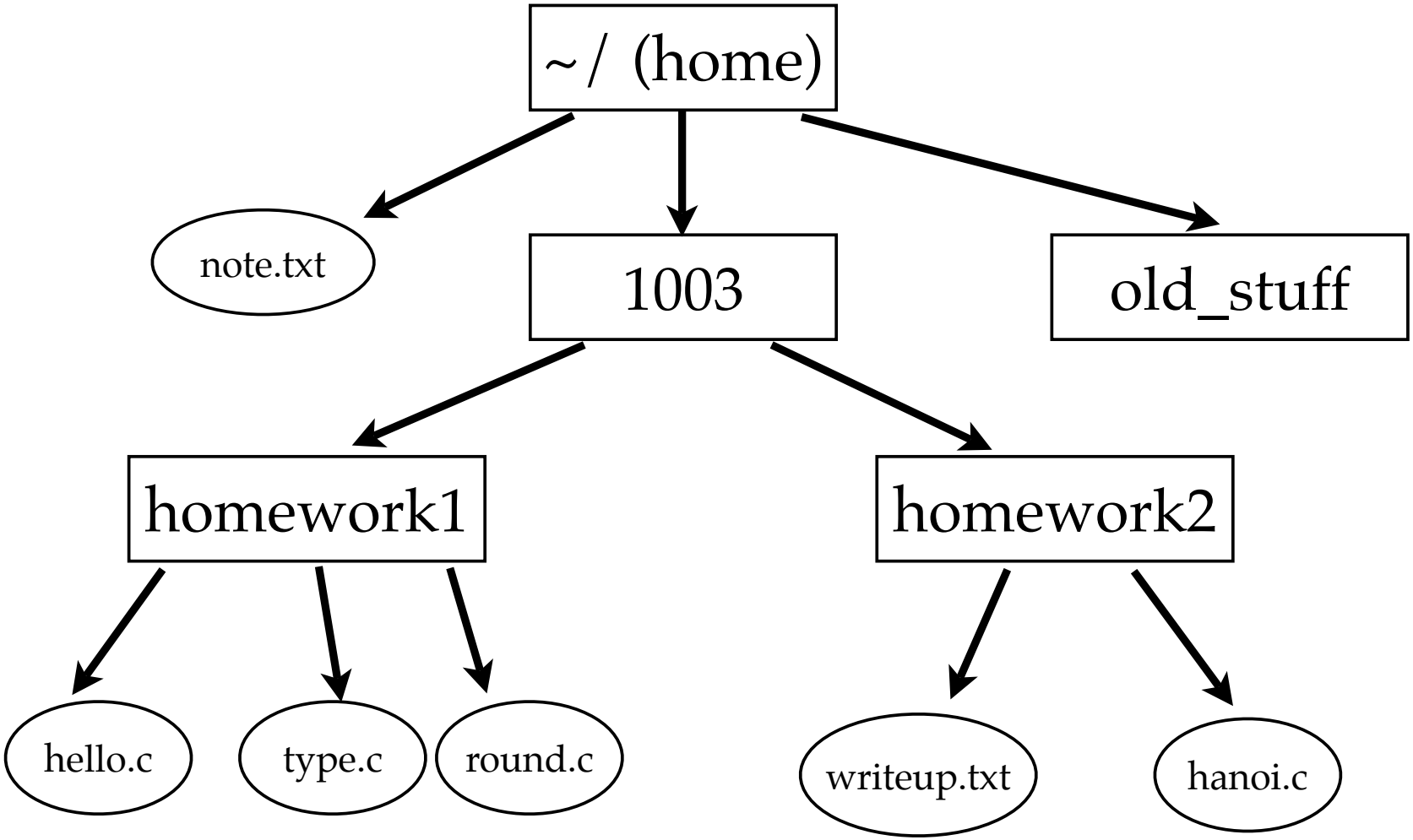
File Systems



Searching

- search_for_file(**current_dir**, **file**)
  - if **file** is in **current_dir**, done!
  - else
    - for each directory **sub_dir** in **current_dir**
      search_for_file(**sub_dir**, file)

Searching


~/ (home)
note.txt
1003
old_stuff
homework1
homework2
hello.c
type.c
round.c
writeup.txt
hanoi.c

Searching

- Recursive search algorithm defines a deterministic order to search.
- (This method is called Depth First Search)

struct

- Often we want to use sets of variables together.
- Example: contact info of a business
  - ```
    float longitude, latitude;
    char address[256], name[128];
    int phone;
    ```
- Storing like this become cumbersome when we have more than one business.
    ```

struct

- C comes with a special type, called a `struct`

```
struct business {
    float longitude, latitude;
    char address[256], name[128];
    int phone;
};
```

- Access these values using . (period)

```
struct business wachovia;
wachovia.phone = 8005551234;
strcpy(wachovia.name, “Citigroup”);
```

struct

- **struct** is short for **data structure**.
- Each value inside a **struct** is called a **field**
- We can treat **struct**s as any variable
  - functions return structs or take as arguments
  - Arrays of structs (simple databases)

```
struct business fortune[500];
```

union

- In rare cases, we need to have a data structure that can have multiple types.
- In a sense, we want to override C's feature of having types.
- ```
  union value {          /* variables of this type */
      long int i_value; /* can be used as either  */
      float f_value;    /* long int or float */
  };
  ```
  ```

union

```
union value {          /* variables of this type */
    long int i_value; /* can be used as either  */
    float f_value;    /* long int or float */
};
```

```
union value data;
data.i_value = 10;
```

- Union allows us to “legally” give variables multiple types.
- Usually unnecessary on modern computers.

typedef

- We can also define custom types using **typedef**
- ```
  typedef int number;
  ```
- Now

  ```
  number x;
  ```

  is the same as

  ```
  int x;
  ```
- More complicated typedef's become more useful
  ```

typedef

- ```
  struct complex_struct {
       float real;
       float imag;
  };
  typedef struct complex_struct complex;
  ```

- ```
  /* instead of */
  struct complex_struct x,y,z;
  ```

- ```
  /* we can write */
  complex x,y,z;
  ```
  ```

enum

- Sometimes we encode values as int's just because they're one of the most basic types.

```
const int FIRSTYEAR = 0, SOPHOMORE = 1,
    JUNIOR = 2, SENIOR = 3, GRAD = 4;
```

- This is convenient so we can write things like:

```
bert.class = GRAD;
/* instead of */
bert.class = 4;
/* which can be confusing. */
```

enum

- **enum** does this cleanly, producing a new type
- ```
  enum class { FIRSTYEAR, SOPHOMORE, JUNIOR, SENIOR,
      GRAD };
  ```
- ```
  class bertsClass = GRAD;
  ```

Programming Tips

- Understand the problem and the solution before you code.
  - Be able to express in English how you plan on solving the problem
- Test incrementally
  - Write your code in pieces and test each piece as you build your program.

Programming Tips

- If you don't know how to solve part of a problem, abstract it and work on the rest of it. **Isolate** the part you're stuck on.
- Use printf()'s to check if variables are what you expect them to be.
- Comment your code!

Reading

- Seriously, start the homework
- Practical C Programming, Chapter 12