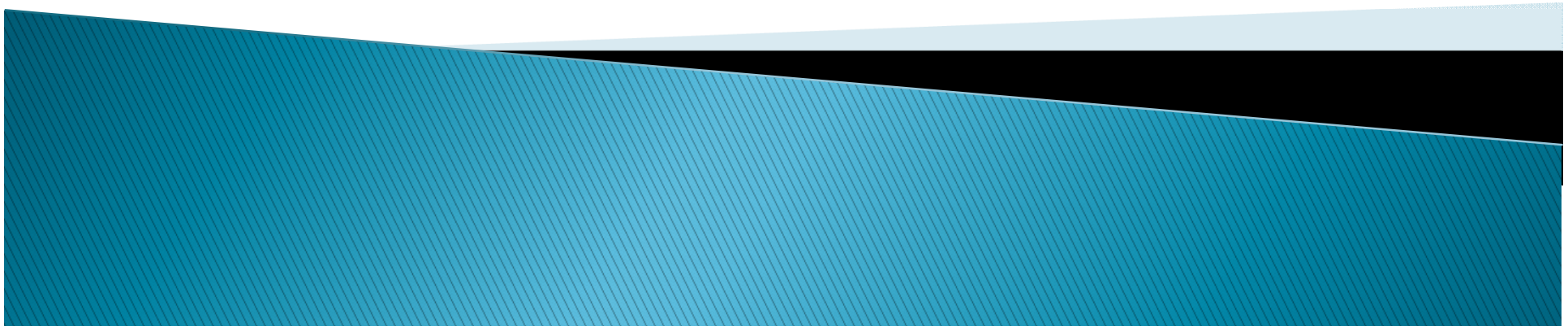
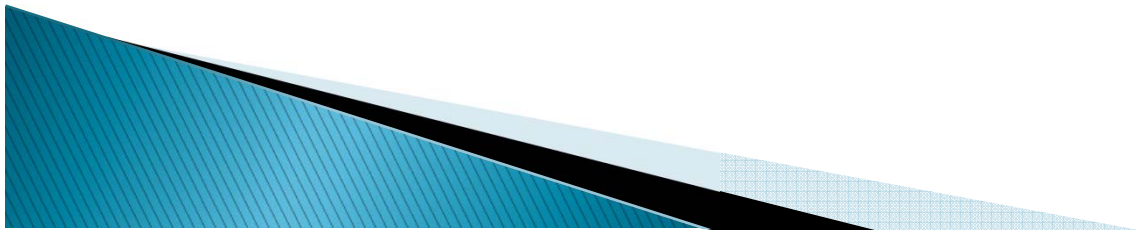
CS502 Project 2

Man Wang
CS502 TA
wang80@purdue.edu



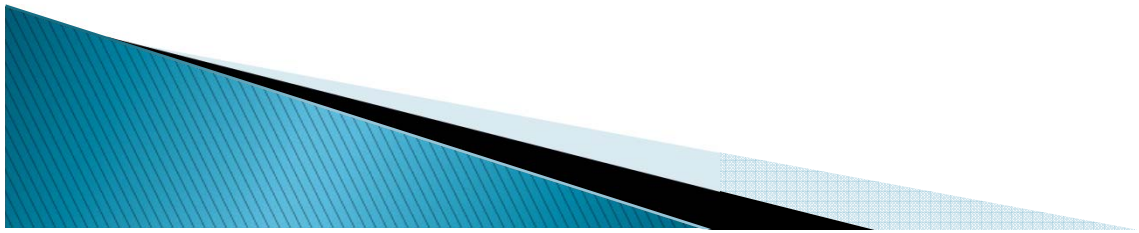
Outline

- Project Overview
- Project Requirements
- How to Start
- Hints and Suggestions
- How to submit
- Resources



Project Overview

- Goal: Implement an interprocedural analysis on uninitialized variables
  - Uninitialized variables :“Used” before “defined”
  - Dataflow analysis
    - Identify program points where uninitialized uses may *potentially* take place
    - Conservative
  - GCC 4.7.0 GENERIC



Project Requirements

- Scope
  - The same subset of C language as in Project1
  - Uninitialized uses of *scalar* variables
  - Limited form of alias analysis

Handle:

```
void init(int *b) {
*b = 1;
}
void main(){
    int a;
    init(&a);
    printf("%d\n", a);
}
```

Condition_1:The address of a scalar variable may be passed to a function

NOT Handle:

```
void main(){
    int a;
    int *p = &a;
     *p = 1;
    printf("%d\n", a);
}
```

ASSUME: A pointer never points to any local variable declared in the same function, nor does it point to any global variable

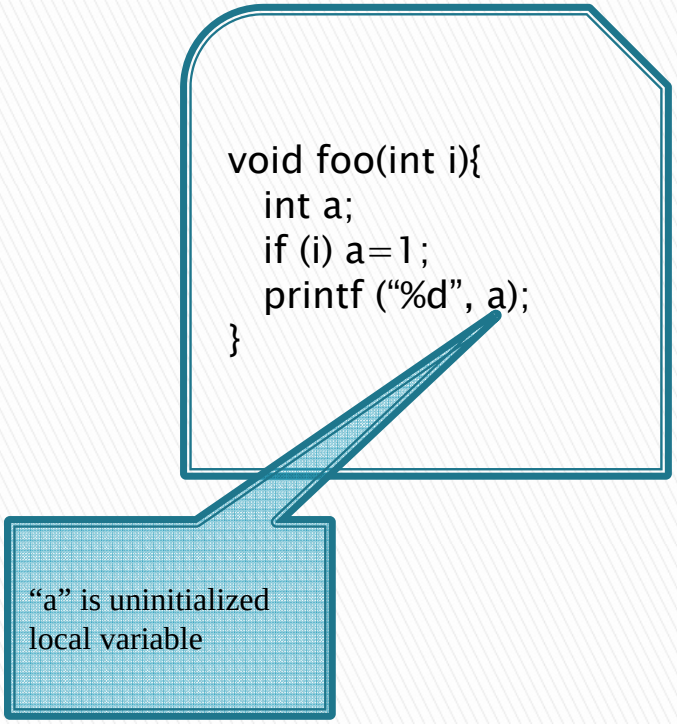
NOT Handle:

```
void init(int *b, int *c) {
*b = 1;
 *c = 2; }
void main(){
    int a;
    init(&a, &a);
    printf("%d\n", a);
}
```

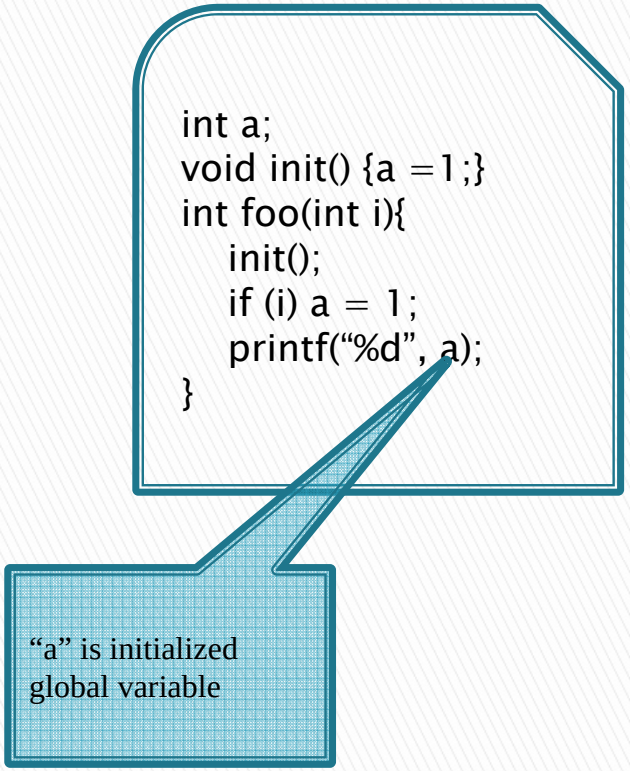
ASSUME: Different pointer variables never point to the same memory addresses.

Project Requirements

- Uninitialized local variable



- Uninitialized Global variable



Project Requirements

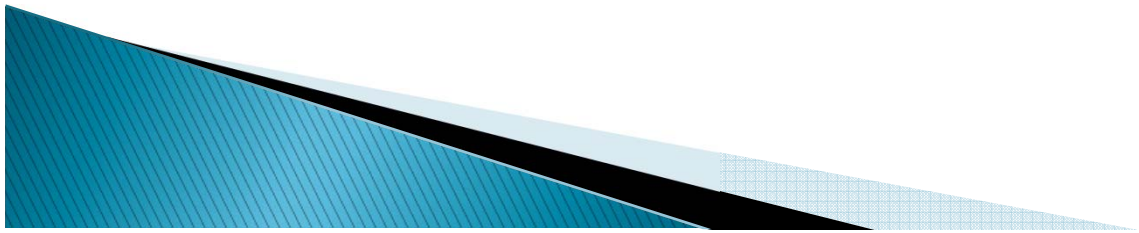
- Output
  - Name : “output.txt”
  - Format

Input:

```
void foo(int i){
   int a;
   if (i) a=1;
   printf (“%d”, a);
}
int main(){
   int x;
   scanf (“%d\n”, &x);
   foo(x);
}
```

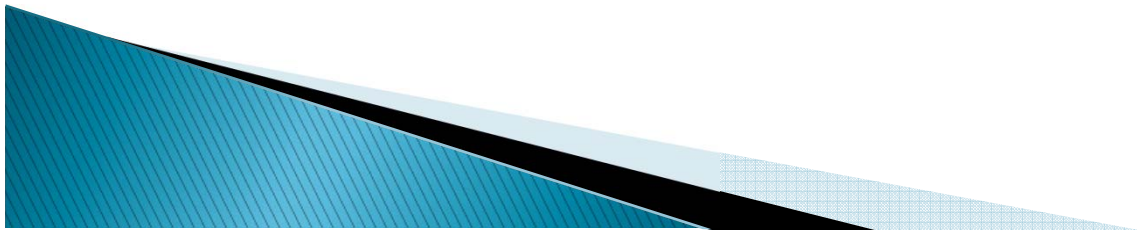
Output:

```
foo: a
```



How to Start

- Download
  - Directory: /u/data/u3/cs502/Fall12/CS502
  - File: cs502_fall12_p2.tar.gz
    - Install
      - working directory
      - Makefile, template file
    - Gcc-4.7.0_src
      - softlink
- Work under your SCRATCH directory
  - $ cd ~/scratch



Hints and Suggestions

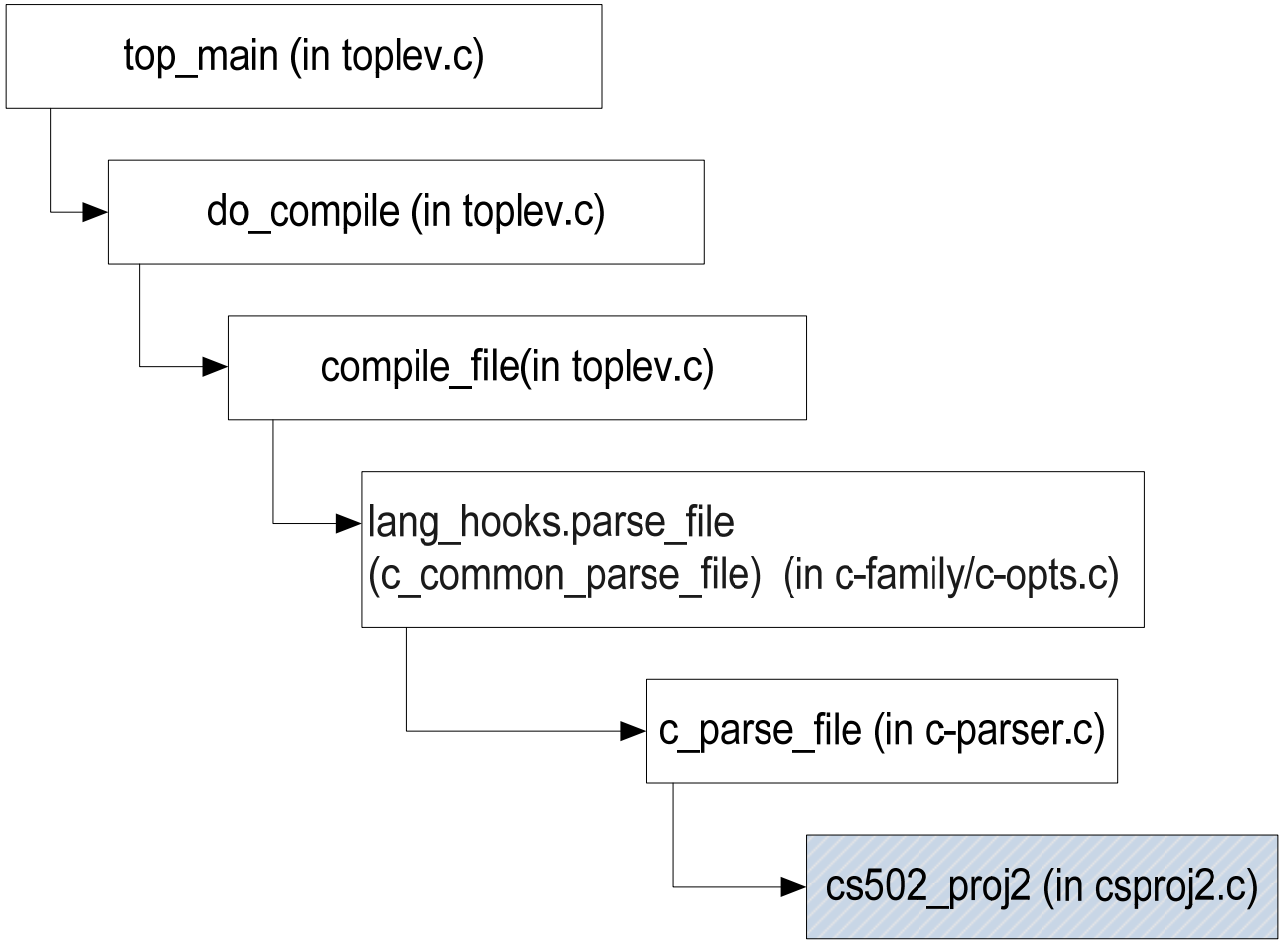
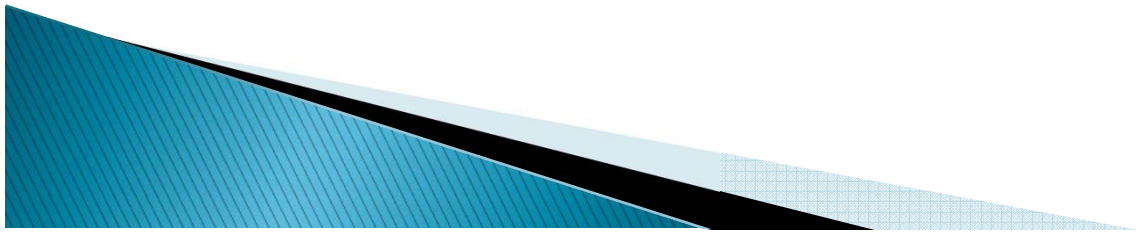


Fig. Where Project 2 is invoked

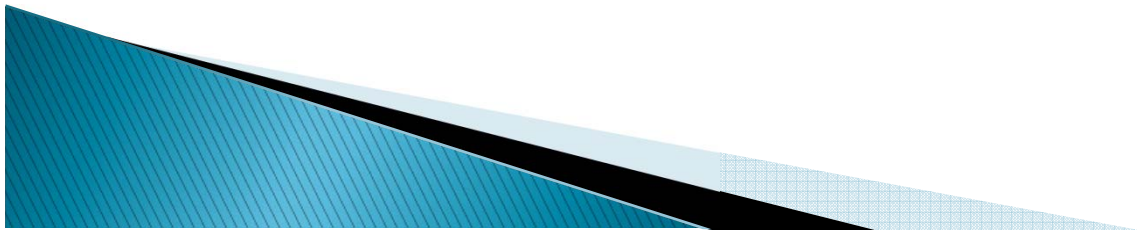
Hints and Suggestions

- **One Suggested Algorithm**
  - Similar to live variable analysis
  - Backward vs. Forward
  - Interprocedural analysis
    - Call Graph
    - Information recorded by each function



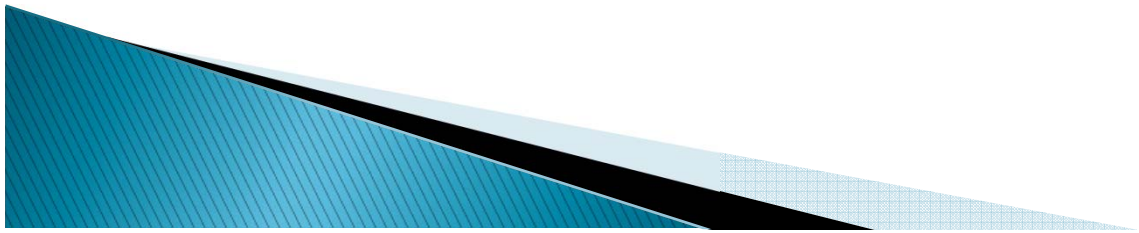
How to submit

- NOTE1
  - Make Clean
- NOTE2
  - NO softlink
- NOTE3
  - README



Resources

- GCC 4.7 internal manual, Tree:
  - http://gcc.gnu.org/onlinedocs/gccint/index.html#toc_GENERIC
- GCC 4.7 Source code
  - linked from “gcc-4.7.0_src”.
- Textbook
  - Data-flow analysis



Q & A