The GNU Debugger

-A GNU Project Initiative.

Presented to

The School of Computer and Information Sciences,

University of Hyderabad

A Data Structures and Programming Lab Class Presentation

Created under the guidance of

Prof. Anjeneya Swami Kare

By Group 2:

Deme Sai Kiran

Anubhab Chakraborty

Shruti Singhania

Srijita Majumdar

Arkaprabha Basu

Kranthi Kiran Akumarthi

M. Tech. Computer Science

Batch of 2021-23

Introduction:
A Brief History of the notorious 'Bugger family':
Bugs, Debugging & Debuggers.

Bugs

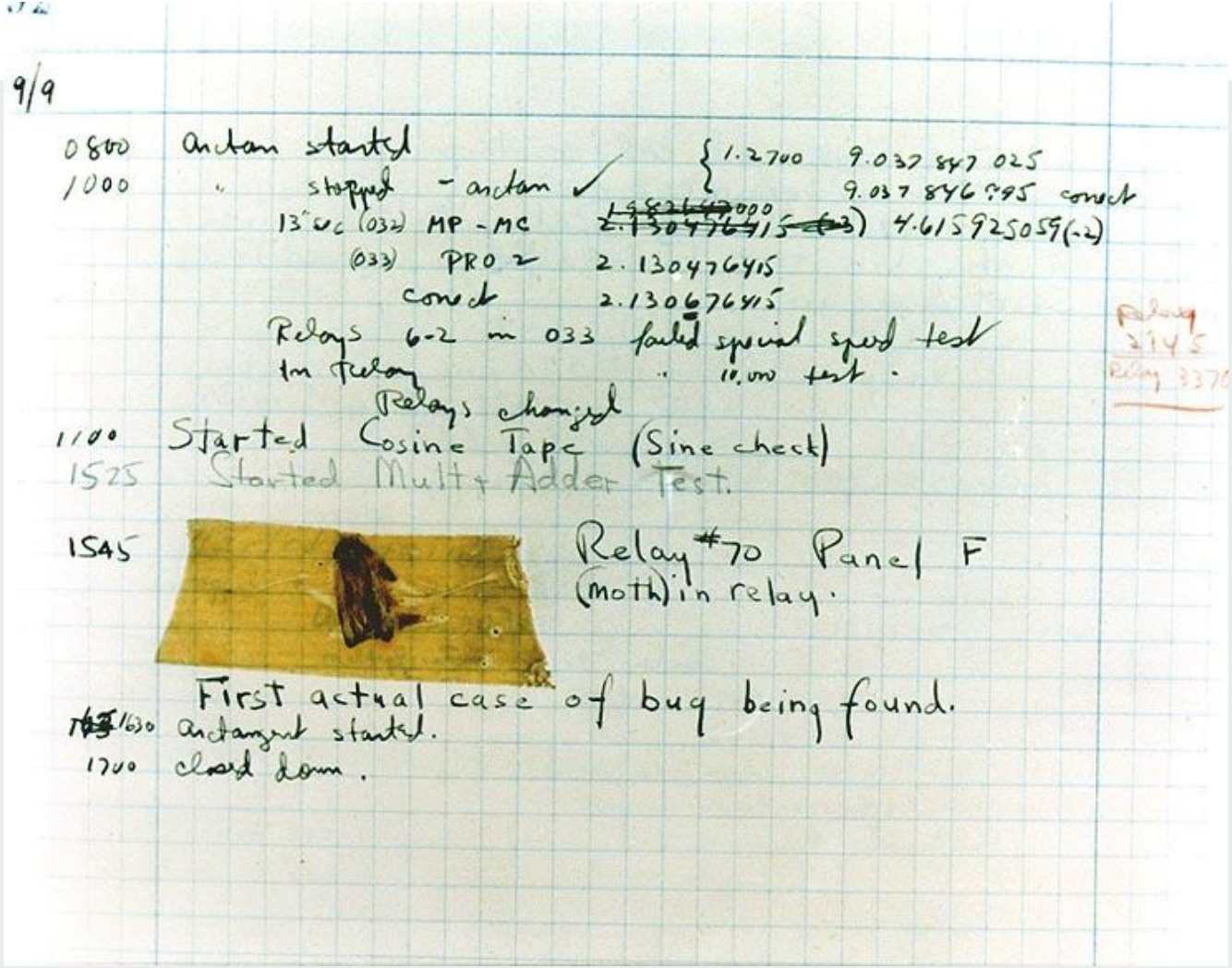
- Flaw in a software program that deviates it from it's required functionality, security or output.
- Term first coined by Edison, comes from a literal rendition of Admiral Grace Hopper who discovered a moth stuck in a relay of the MARK II computer, impending operation in 1947 (coincidence?).

Debugging

- Process of finding and resolving bugs within a computer, software program or in any functional system.

Debugger

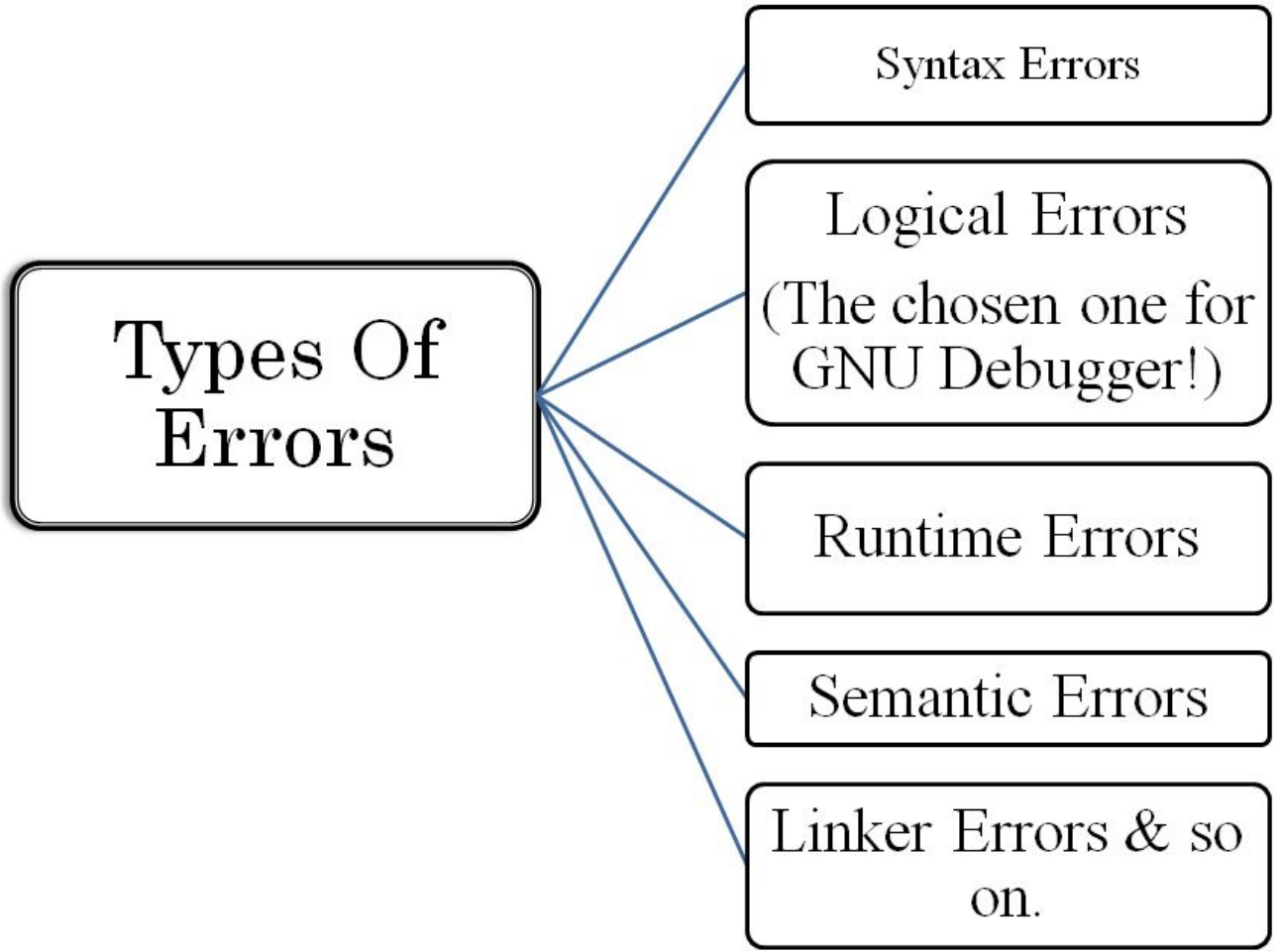
- Runs the target program under controlled conditions, allowing the programmer to track its working operations and monitor changes in computer resources indicating a malfunctioning code.



History's 1st 'Bug'.

Location & Date:
Harvard University,
September 9, 1947.

Courtesy : National Geographic & the Naval Surface Warfare Centre, Dahlgren, Virginia.


Types Of Errors
Syntax Errors
Logical Errors
(The chosen one for GNU Debugger!)
Runtime Errors
Semantic Errors
Linker Errors & so on.

Breakpoint

- Breakpoints are markers that can be set at any executable line of code for intentionally pausing or stopping at a particular place in the program.
- When program execution reaches a breakpoint, execution pauses allowing for the examination of variable values to help determine whether a logic error exists.
- NOTE - Attempting to set a breakpoint at a line of code that is not executable (such as a comment) actually sets the breakpoint at the next executable line of code in that function.

Convenience variables

- Convenience variables are temporary variables created by the debugger that are named using a dollar sign followed by an integer.
- They can be used to perform arithmetic operation and evaluate boolean expressions.
- When the print command is used, the result is stored in a convenience variable such as $1.
- NOTE - Printing the value of $1 creates a new convenience variable - $2

Basic GDB Commands

- The GNU GDB offers a CUI to debug C programs in the Linux environment.
- Among the numerous commands GDB has to perform a variety of debugging operations, in this presentation, we shall go through some of the most common ones.
- We shall try to understand the working of the following GDB commands - gdb, run, break, continue, print, delete, quit, set, list, help, step, finish, info break, watch, next
- The instructions are divided into a few sections for easier understanding and in the first section we will see how to start, run and stop the GDB after compiling the C program.
- The second section covers the creation and manipulation of breakpoints in GDB.
- The final section introduces the instructions to control the execution flow of C programs.

Basic GDB Commands - *Section I*

- Compiling for debugging
  - To enable debugging for a program compile using **gcc -g program.c**
- Starting the debugger
  - Use **gdb** command before running the program like **gdb ./a.out** (a.out can be replaced by the name of the executable file generated after compilation)
- Inserting breakpoints
  - Breakpoint can be set at specific lines of code by using the **break** command followed by the line number for e.g. **break 14**
- Running the program
  - Once the debugger is started, start execution of the C program using **run** command

Basic GDB Commands - *Section I*

- Examining a variable's value
  - To display the current value of a variable the **print** command is used with the variable name or convenience variables for e.g. **print value1** or **print $1**
- Resuming execution after breakpoint
  - The **continue** command resumes execution flow till the next breakpoint is reached
- Stopping the debugger
  - The **quit** command causes the debugger to terminate

Basic GDB Commands - *Section II*

- Using **info break** command:
  - It displays a table of all breakpoints with its properties listed below.
    - Num, Type, Disp, Enb, Address, What.
- Removing a Breakpoint:
  - Using **delete** command we can delete a particular breakpoint.
  - Type delete followed by space and the breakpoint number.
- Example:
  - (gdb) **info break**

Num	Type	Disp	Enb	Address	What
1	breakpoint	keep	y	0x080478e3	in main at text.c:15

    breakpoint already hit 1 time.
  - (gdb) **delete 1**
  - (gdb) **info break**

    No breakpoints or watchpoints.

Evaluating Expressions & Modifying Values

- After entering the break mode we can evaluate both Arithmetic and Boolean expressions.
- We can Evaluate expressions with print command to find logical error.
  - Arithmetic expression:

    Ex: (gdb) **print number1 + 20** (consider value of number1 is 20 in the program)

    $1 = 40
  - Boolean expressions:

    Ex: (gdb) **print number1** == 40

    $2 = 1

  Note: print command will add convenience variable to Value history.
- **Modifying Values:** Values of variables can be changed using **set** command.
- Set command will not add to Value history.
  - Ex: (gdb) set number1 = 38

    (gdb) print number1

    $3 = 38

Basic GDB Commands - *Section III*

- Allows execution of a program / function Line By Line to find and fix the errors.
- Using **Step** Command:
  - Executes the next statement of the program
  - If the next statement is a function, returns to the function call
  - Stops inside a called function
- Using **Finish** Command:
  - Evaluates the remaining statement and returns to the function call section
- Using **Next** Command:
  - Step returns to the called function but Next executes it at nearly full speed
  - Stops only at the next line in the current function
- Using **Watch** Command:
  - Tracks a data member (Variable), it changes, Watch will notify.
    - `watch var1`
  - When var1 changes, debugger enters a break mode and notifies.
- More commands - Continue, quit

Keeping Short

1. GNU (GNU's not Unix) includes a debugger which empowers tracking the program performance and resolve the errors
2. Enter `gdb` to start the GNU Debugger, `run` command to run a program
3. Breakpoints are included to pause the program at any line of the code
4. Commands Discussed: `break, continue, print, delete, quit, set, list, help, step, finish, info break, watch, next, set, info break, delete, step, finish, next and watch`