GDB Basics — Beginner Friendly Notes

GDB (GNU Debugger) is a powerful tool for debugging **C/C++ programs**. It allows you to inspect, control, and debug your program step by step.

1️⃣ Installing GDB

```
sudo apt install gdb
```

Check installation:

```
gdb --version
```

2️⃣ Compiling for Debugging

When compiling your program, always use the `-g` flag to include debug symbols.

```
gcc -g -o example example.c
```

3️⃣ Common GDB Commands

Command	Action
`gdb ./program`	Start GDB with your program
`run` or `r`	Run the program inside GDB
`break <line/function>` or `b`	Set breakpoint at line or function
`next` or `n`	Execute next line (step over function calls)
`step` or `s`	Step into a function
`continue` or `c`	Continue execution until next breakpoint
`print <variable>` or `p`	Print value of variable
`info locals`	Show all local variables in current scope

Command	Action
`backtrace` or `bt`	Show call stack
`list` or `l`	Show source code around current line
`quit` or `q`	Exit GDB

4️⃣ Example 1: Debugging a Simple Division Error

File: `example.c`

```c
#include <stdio.h>

int main() {
    int a = 10;
    int b = 0;
    int c;

    c = a / b; // Error: division by zero

    printf("Result: %d\n", c);
    return 0;
}
```

5️⃣ Example 2: Debugging a Function with Array Out-of-Bounds

File: `array_example.c`

```c
#include <stdio.h>

void print_element(int arr[], int index) {
    printf("Element at %d: %d\n", index, arr[index]);
}

int main() {
    int numbers[5] = {1, 2, 3, 4, 5};
    int i;

    for (i = 0; i <= 5; i++) { // Error: goes out of bounds
        print_element(numbers, i);
    }
```

```
    return 0;
}
```

6️⃣ Example 3: Debugging a Larger Program with Multiple Errors

File: `big_example.c`

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

void print_string(char *str) {
    printf("String: %s\n", str);
}

int sum_array(int arr[], int size) {
    int sum = 0;
    for (int i = 0; i <= size; i++) { // Error: should be < size
        sum += arr[i];
    }
    return sum;
}

int main() {
    int nums[3] = {10, 20, 30};
    char *msg = NULL;

    print_string(msg); // NULL pointer error

    int total = sum_array(nums, 3);
    printf("Total: %d\n", total);

    int x = 5;
    int y = 0;
    int z = x / y; // Division by zero
    printf("Division result: %d\n", z);

    return 0;
}
```

7️⃣ Example 4: Debugging a 100-line Program

File: `student_management.c` - This is a simplified student management system with multiple errors including array overflow, NULL pointer usage, and invalid input.

```
#include <stdio.h>
#include <string.h>

typedef struct {
    char name[50];
    int age;
    float marks;
} Student;

void print_student(Student *s) {
    printf("Name: %s, Age: %d, Marks: %.2f\n", s->name, s->age, s->marks);
}

float average_marks(Student students[], int n) {
    float sum = 0;
    for (int i = 0; i <= n; i++) { // Error: should be < n
        sum += students[i].marks;
    }
    return sum / n;
}

int main() {
    Student students[5];
    int i;

    // Initialize students with some data
    for (i = 0; i <= 5; i++) { // Error: goes out of bounds
        printf("Enter name, age, marks for student %d: ", i+1);
        scanf("%s %d %f", students[i].name, &students[i].age,
&students[i].marks);
    }

    // Print all students
    for (i = 0; i < 5; i++) {
        print_student(&students[i]);
    }

    float avg = average_marks(students, 5);
    printf("Average Marks: %.2f\n", avg);

    // Deliberate NULL pointer error
```

```
    Student *null_student = NULL;
    print_student(null_student);

    // Division by zero error
    int x = 10, y = 0;
    int z = x / y;
    printf("Division result: %d\n", z);

    return 0;
}
```

Debugging Steps: 1. Compile with `-g`. 2. Start GDB: `gdb ./student_management` 3. Set breakpoints at `main`, `print_student`, and `average_marks`. 4. Run program: `run`. 5. Step through loops and functions with `next` or `step`. 6. Detect and fix errors: - Array index exceeding size. - Null pointer usage. - Division by zero.

8️⃣ Extra Tips

- Use `info breakpoints` to list all breakpoints.
- Use `delete <num>` to remove a breakpoint.
- `watch <variable>` stops execution when the variable changes.
- `set var <variable>=<value>` changes variable values at runtime.
- Use `layout src` in GDB TUI mode to view code and debugger simultaneously.

These notes cover **GDB installation, compilation, basic commands, and debugging examples ranging from simple to 100-line programs with multiple errors** to help beginners learn step-by-step.