Example: Debugging a Simple Kernel Module with KGDB

1. **Create a Simple Kernel Module**:

```c
// simple_module.c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

static int __init simple_init(void) {
    printk(KERN_INFO "Simple Module: Initialization\n");
    return 0;
}

static void __exit simple_exit(void) {
    printk(KERN_INFO "Simple Module: Exit\n");
}

module_init(simple_init);
module_exit(simple_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Your Name");
MODULE_DESCRIPTION("A simple kernel module for debugging example");
```

2. **Compile the Kernel Module**:

   - Create a `Makefile` for the module:

```
obj-m += simple_module.o

all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

   - Run the following command to build the module:

```
make
```

3. **Load the Kernel Module**:

   - Load the module into the kernel:

```
sudo insmod simple_module.ko
```

   - Verify the module is loaded:

```
lsmod | grep simple_module
```

4. **Set Up KGDB**:
   - Compile the kernel with KGDB support:
     - Enable `CONFIG_KGDB` and `CONFIG_DEBUG_INFO` in the kernel configuration.
   - Boot the kernel with the following kernel parameters:

     ```
     kgdboc=ttyS0,115200 kgdbwait
     ```

5. **Connect GDB to KGDB**:
   - On the development machine, run GDB with the `vmlinux` file:

     ```
     gdb vmlinux
     ```

   - In GDB, connect to the target machine:

     ```
     target remote /dev/ttyS0
     ```

   - Set a breakpoint in the kernel module's initialization function:

     ```
     b simple_init
     ```

6. **Debugging the Kernel Module**:
   - Continue execution in GDB:

     ```
     c
     ```

   - When the kernel module is loaded, GDB will break at `simple_init`.
   - You can now inspect variables, memory, and the call stack using GDB commands.

Practical Tips

- **Inspect Variables**: Use the `print` command to check variable values.

  ```
  print my_variable
  ```

- **Step Through Code**: Use `step` to go line by line in the source code.

  ```
  step
  ```

- **Set Breakpoints**: Add breakpoints at specific lines or functions.

  ```
  b function_name
  ```

Example Debugging Session

```
$ gdb vmlinux
(gdb) target remote /dev/ttyS0
Remote debugging using /dev/ttyS0
```

```
0x00000000 in ?? ()
(gdb) b simple_init
Breakpoint 1 at 0x123456: file simple_module.c, line 5.
(gdb) c
Continuing.

Breakpoint 1, simple_init () at simple_module.c:5
5       printk(KERN_INFO "Simple Module: Initialization\n");
(gdb) print some_variable
$1 = 42
(gdb) step
6       return 0;
(gdb) c
Continuing.
```