# Lab4 Integer Stack Kernel Module

Lab4 GitHub

A Linux kernel module that implements a stack for integers with push/pop operations and stack size configuration via ioctl, along with a userspace utility for interacting with it.

## Key Features

1. **Dynamic Memory Management**:
   - Uses `kmalloc` for stack data allocation
   - Dynamically resizes the stack when needed
   - Properly frees memory with `kfree` to prevent memory leaks
2. **Synchronization for Concurrent Access**:
   - Implements reader-writer semaphores for thread-safe access
   - Uses appropriate write locks for stack-modifying operations
   - Module is safe for multi-threaded access
3. **Comprehensive Error Handling**:
   - Returns appropriate error codes (`-ERANGE`, `-EINVAL`, `-EFAULT`, etc.)
   - Checks for null pointers and invalid parameters
   - Handles memory allocation failures
   - Provides detailed kernel logs for debugging
4. **Edge Case Handling**:
   - Empty stack checks
   - Full stack checks
   - Invalid ioctl commands
   - Zero-sized stack prevention

## Building the Kernel Module

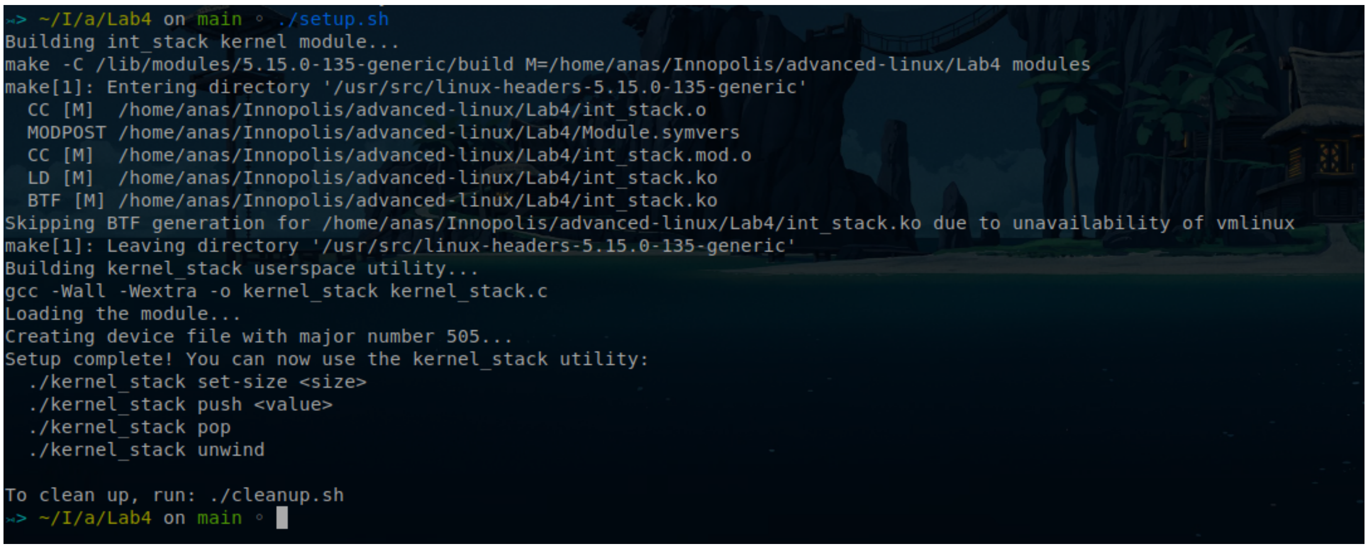
To build the kernel module:

```
# Build the kernel module
make

# Load the module
sudo insmod int_stack.ko

# Check the assigned major number (look for "int_stack: registered with major number X")
dmesg | grep int_stack
```

```
# Create the device file
sudo mknod /dev/int_stack c $MAJOR 0
sudo chmod 666 /dev/int_stack
```

For convenience, you can use the provided setup script:

```
./setup.sh
```



```
-> ~/I/a/Lab4 on main ◦ ./setup.sh
Building int_stack kernel module...
make -C /lib/modules/5.15.0-135-generic/build M=/home/anas/Innopolis/advanced-linux/Lab4 modules
make[1]: Entering directory '/usr/src/linux-headers-5.15.0-135-generic'
  CC [M]  /home/anas/Innopolis/advanced-linux/Lab4/int_stack.o
  MODPOST /home/anas/Innopolis/advanced-linux/Lab4/Module.symvers
  CC [M]  /home/anas/Innopolis/advanced-linux/Lab4/int_stack.mod.o
  LD [M]  /home/anas/Innopolis/advanced-linux/Lab4/int_stack.ko
  BTF [M] /home/anas/Innopolis/advanced-linux/Lab4/int_stack.ko
Skipping BTF generation for /home/anas/Innopolis/advanced-linux/Lab4/int_stack.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-5.15.0-135-generic'
Building kernel_stack userspace utility...
gcc -Wall -Wextra -o kernel_stack kernel_stack.c
Loading the module...
Creating device file with major number 505...
Setup complete! You can now use the kernel_stack utility:
  ./kernel_stack set-size <size>
  ./kernel_stack push <value>
  ./kernel_stack pop
  ./kernel_stack unwind

To clean up, run: ./cleanup.sh
-> ~/I/a/Lab4 on main ◦
```

## Building the Userspace Utility

To build the userspace utility:

```
make -f Makefile.user
```

> Note: This is included in `setup.sh`

## Using the Userspace Utility

The `kernel_stack` utility provides an interface to the kernel module.

### Available Commands

1. **Push an integer onto the stack**:

   ```
   ./kernel_stack push VALUE
   ```

2. **Pop an integer from the stack**:

   ```
   ./kernel_stack pop
   ```

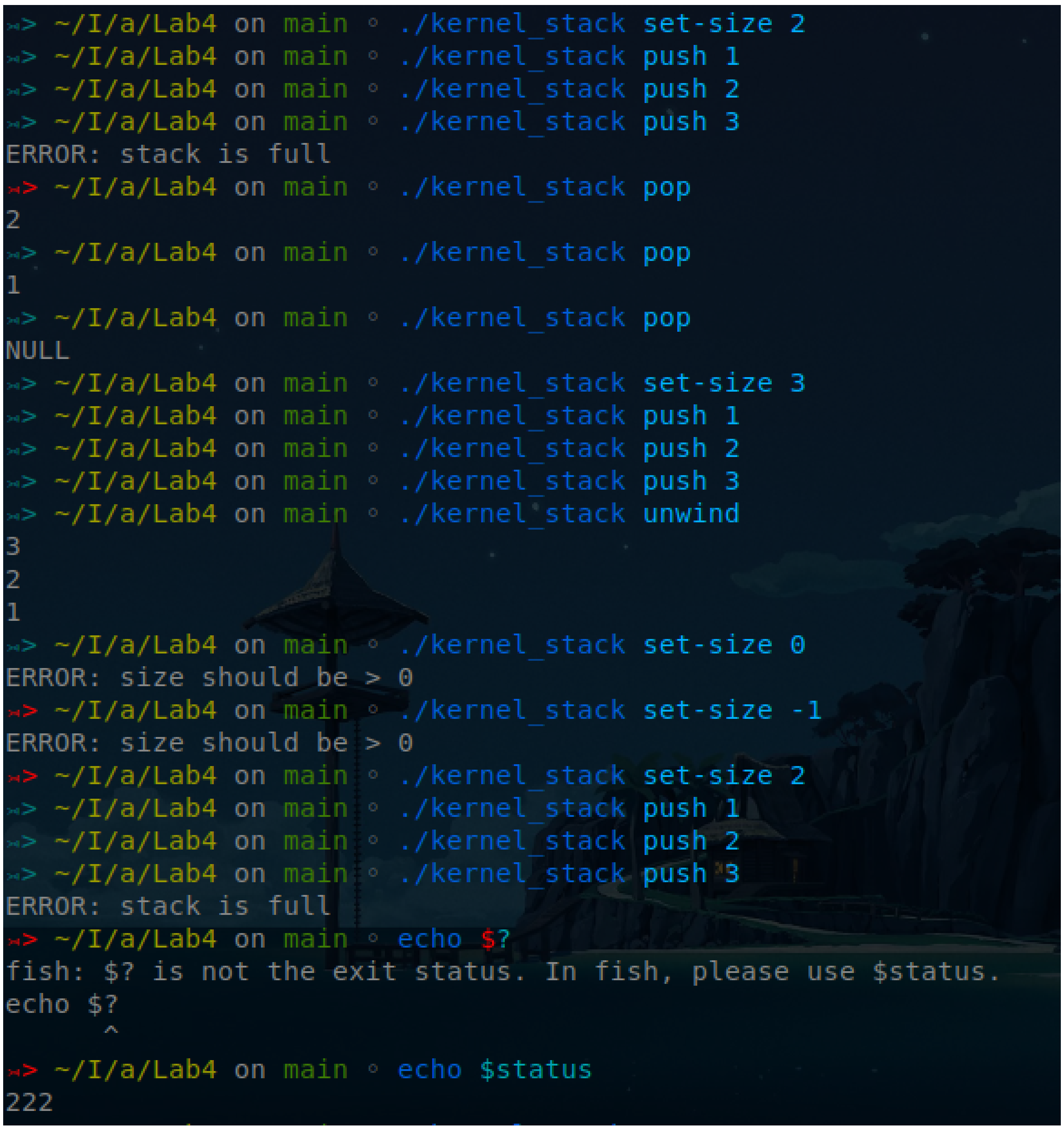
3. **Unwind the stack (pop and print all values)**:

   ```
   ./kernel_stack unwind
   ```

4. **Set the maximum stack size**:

   ```
   ./kernel_stack set-size SIZE
   ```

## Examples



```
~> ~/I/a/Lab4 on main ◦ ./kernel_stack set-size 2
~> ~/I/a/Lab4 on main ◦ ./kernel_stack push 1
~> ~/I/a/Lab4 on main ◦ ./kernel_stack push 2
~> ~/I/a/Lab4 on main ◦ ./kernel_stack push 3
ERROR: stack is full
~> ~/I/a/Lab4 on main ◦ ./kernel_stack pop
2
~> ~/I/a/Lab4 on main ◦ ./kernel_stack pop
1
~> ~/I/a/Lab4 on main ◦ ./kernel_stack pop
NULL
~> ~/I/a/Lab4 on main ◦ ./kernel_stack set-size 3
~> ~/I/a/Lab4 on main ◦ ./kernel_stack push 1
~> ~/I/a/Lab4 on main ◦ ./kernel_stack push 2
~> ~/I/a/Lab4 on main ◦ ./kernel_stack push 3
~> ~/I/a/Lab4 on main ◦ ./kernel_stack unwind
3
2
1
~> ~/I/a/Lab4 on main ◦ ./kernel_stack set-size 0
ERROR: size should be > 0
~> ~/I/a/Lab4 on main ◦ ./kernel_stack set-size -1
ERROR: size should be > 0
~> ~/I/a/Lab4 on main ◦ ./kernel_stack set-size 2
~> ~/I/a/Lab4 on main ◦ ./kernel_stack push 1
~> ~/I/a/Lab4 on main ◦ ./kernel_stack push 2
~> ~/I/a/Lab4 on main ◦ ./kernel_stack push 3
ERROR: stack is full
~> ~/I/a/Lab4 on main ◦ echo $?
fish: $? is not the exit status. In fish, please use $status.
echo $?
     ^

~> ~/I/a/Lab4 on main ◦ echo $status
222
```

> Note:
>
> - The status is `222` not `-34` is related to how shell return codes.

> - Exit codes are 8 bits (0-255) so `-34` in two's complement become `222` when interpreted as unsigned
> - I couldn't solve the issue and tried in different ways, I hope if you can suggest how to resolve this in the comments.

## Implementation Details

### Kernel Module

- The kernel module (`int_stack.ko`) implements a dynamically resizable stack of integers.
- It exposes a character device that can be accessed through standard file operations.
- Memory for the stack is allocated dynamically and grows/shrinks as needed.
- The module utilizes reader-writer semaphores for synchronization, allowing multiple readers but exclusive writers.
- Proper error codes are returned for all edge cases and error conditions.

### Userspace Utility

- The utility (`kernel_stack`) provides a user-friendly interface to interact with the module.
- It handles all communication with the device file and formats output appropriately.
- It translates error codes to user-friendly messages.

## Module File Operations

- **open()**: Initializes the stack if not already done
- **release()**: Updates module reference count
- **read()**: Pops an integer from the stack
- **write()**: Pushes an integer onto the stack
- **ioctl()**: Configures the maximum stack size

## Error Codes

The module follows standard kernel error codes:

- `-ERANGE`: Stack is full (on push)
- `-EINVAL`: Invalid parameter or operation
- `-EFAULT`: Failed to copy data between kernel and user space
- `-ENOMEM`: Memory allocation failed
- `-ENOTTY`: Invalid ioctl command

## Cleanup

```
# Remove the device file
sudo rm /dev/int_stack

# Unload the module
sudo rmmod int_stack

# Clean up build artifacts
```

```
make clean
make -f Makefile.user clean
```

Or simply run:

```
./cleanup.sh
```