Modified xv6 System Calls Project

Team Members:

P Naga Sripada - CS22B1018
Varshitha Masaram - CS22B1071
G Satya Priya - CS22B1012
N Priyadarshini - CS22B1009

Contents

1 Objective

This project involves modifying the xv6 operating system to implement new functionalities by creating or modifying system calls. The tasks include: Setting up the development environment for xv6. Analyzing existing system calls in xv6. Modifying or creating at least five new system calls. Demonstrating these functionalities using user programs. Properly documenting the changes and execution results.

2 Environment Setup

The environment setup is the first step in working with the xv6 operating system. It involves preparing the necessary software and tools to build, run, and test the system. This setup includes installing Ubuntu on a virtual machine, cloning the xv6 repository, and configuring the system to compile and run the xv6 kernel. The following steps outline the detailed process for setting up the development environment.

- Install Ubuntu via VirtualBox or any other virtualization software.
- Clone the xv6 repository using:
  `git clone https://github.com/mit-pdos/xv6-public.git`
- Navigate to the xv6 directory:
  `cd xv6-public`
- Build the xv6 operating system:
  `make`
- Clean the build using:
  `make clean`
- Run xv6 in QEMU:
  `make qemu`

3 Implemented Modifications and New System Calls

In this project, several modifications were made to the existing system calls in the xv6 operating system, along with the introduction of new system calls to extend its functionality. These modifications were aimed at improving the process management, signal handling, and inter-process communication (IPC) mechanisms within the kernel. The system calls were modified or created to support more complex behaviors, such as multi-level process creation, enhanced exit handling, custom signal handling, process-specific wait functionality, and priority-based pipes.

Each modification was carefully implemented to ensure that the new behavior seamlessly integrates with the existing system. These changes offer more flexibility and control over processes, signals, and resources within the operating system. Below are the descriptions of each modification and the newly created system calls.

The following sections describe the system calls added or modified:

3.1 1. Multi-Level Process Creation (`fork()`)

Description: Modified the `fork()` system call to support an extended syntax for multi-level process creation. For example, `fork(3)` creates 3 child processes for the parent. Extending this, `fork(3,1,2,0)` creates 3 child processes for the parent, and further assigns 1, 2, and 0 children to each of the respective children.

Implementation Steps: The implementation involved modifying the kernel code to accept additional parameters for process creation. A recursive approach was implemented to allow the system call to create child processes at multiple levels based on the input parameters. The changes to the kernel code ensured that each child could further spawn its own children based on the values specified. This required modifying the process table to handle the new child processes and managing the creation of processes within the kernel.

Logic: The logic for multi-level process creation begins by parsing the input parameters. The first parameter specifies the number of child processes the parent will create. Subsequent parameters define how many children each of those child processes will have. For example `fork(3, 1, 2, 0)` means the parent creates 3 child processes, the first child will have 1 child, the second will have 2 children and the third will have none. The kernel iterates over the parameters, recursively creating child processes and assigning children to each respective process according to the parsed parameters.
Usage:

```
int result = fork(3, 1, 2, 0);
```

3.2 2. Extended Exit Functionality (`exit2()`)

Description: The exit() system call was modified to create a new system call, exit2(), which enhances the child cleanup logic when a process terminates. The modification ensures that when a parent process exits, it properly handles its child processes by marking them as killed and waking any that are sleeping. The system waits for child processes to enter the ZOMBIE state before cleaning them up. Additionally, orphaned children are reassigned to the initproc, and the parent process itself is transitioned into the ZOMBIE state once it has completed the cleanup.

Implementation Steps: The implementation involves iterating over the child processes of the parent to handle their termination. For each child, resources are released, and orphaned children are reassigned to the initproc to ensure they are properly managed. The parent process itself performs cleanup, removing references to any remaining children and ensuring that the process is marked as a ZOMBIE after its tasks are completed.

3.3 3. Modified Signal Handling (`SIGINT`)

Description: The `SIGINT` signal was modified to implement a toggle mechanism that alternates between pausing and resuming a process when `Ctrl+C` is pressed. This new functionality allows for better control over running processes, enabling users to temporarily halt a process and later resume it without terminating it. The process transitions between suspended and running states based on consecutive `SIGINT` signals, providing more flexibility in process management.

Logic: When the first `SIGINT` is received, the process is transitioned into a suspended state, effectively pausing its execution. Upon receiving the next `SIGINT`, the process is resumed by switching it back to the running state. This mechanism relies on toggling between the two states each time `SIGINT` is triggered, offering a simple yet effective way to pause and resume process execution.

Usage:

```c
void custom_handler(int sig) {
    static int paused = 0;
    if (paused) {
        printf("Resuming process...\n");
    } else {
        printf("Pausing process...\n");
    }
    paused = !paused;
}
int main() {
    signal(SIGINT, custom_handler);
    while (1) ;
    return 0;
}
```

3.4 4. Wait for Specific Process (`waitpid()`)

Description: The `waitpid()` system call was implemented to allow a parent process to wait for a specific child process to finish. It enables the parent to search for a child process with the specified Process ID (`PID`). If the child has completed execution and is in the ZOMBIE state, its exit status is retrieved and the resources are cleaned up. If the child has not yet terminated, the parent is blocked until the child transitions to the ZOMBIE state.

Usage:

```c
int status;
int result = waitpid(child_pid, &status);
```

3.5 5. Priority Pipes

Description: The pipe() system call was enhanced to support priority-based message delivery. A priority field was added to the pipe structure, allowing messages to be tagged with priority levels. When multiple messages are written to the pipe, they are sorted in descending order of priority. The messages are then read in the order of their priority, ensuring that higher-priority messages are processed first.

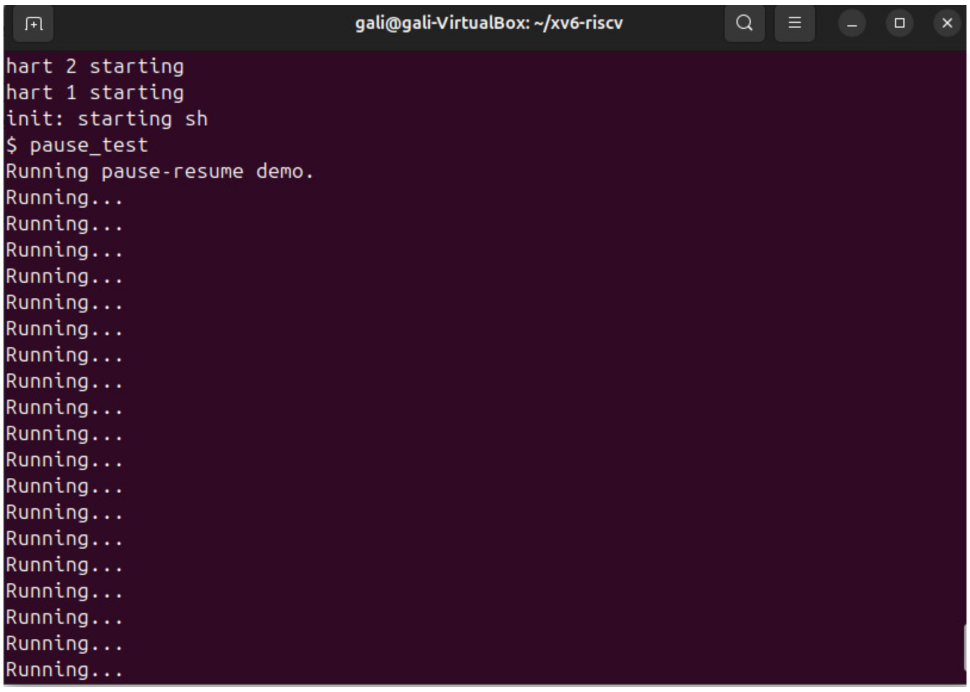
Logic: The logic for the priority-based pipe system involves modifying the pipe structure to include an additional field for priority. When a message is written to the pipe, its priority is stored along with the message. A sorting mechanism is then employed within the pipe to reorder the messages before they are read, ensuring that messages with higher priority are processed first. This ensures efficient message delivery based on priority levels.

Usage:

```
write(pipe_fd, "Message", priority);
read(pipe_fd, buffer);
```

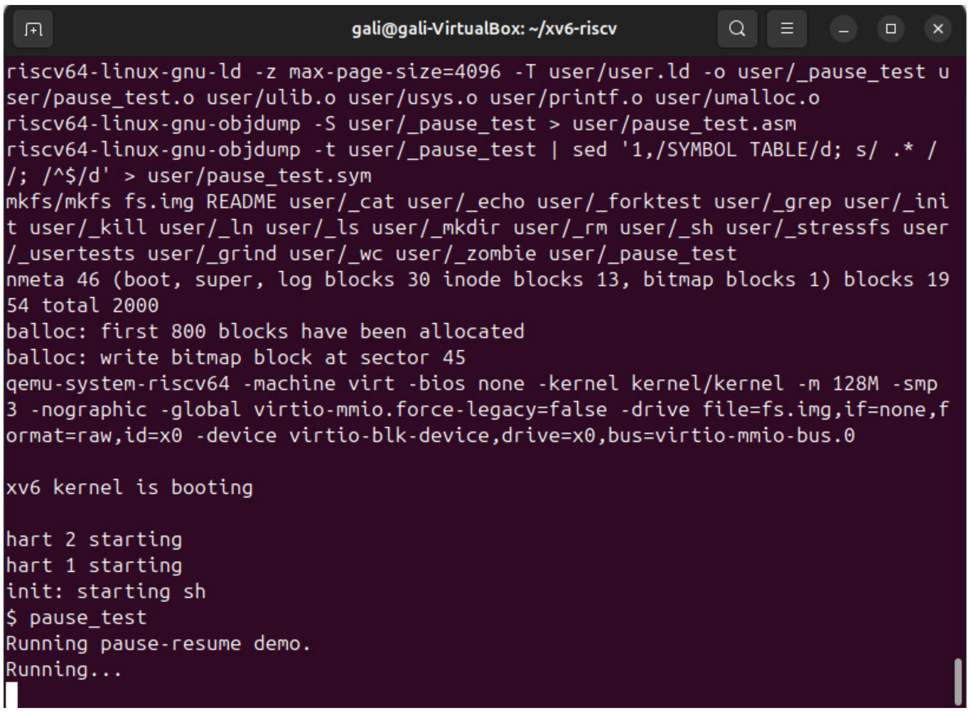
4 Demonstration with User Programs

4.1 SIGINT



```
hart 2 starting
hart 1 starting
init: starting sh
$ pause_test
Running pause-resume demo.
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
```

Figure 1: Output of SIGINT



gali@gali-VirtualBox: ~/xv6-riscv

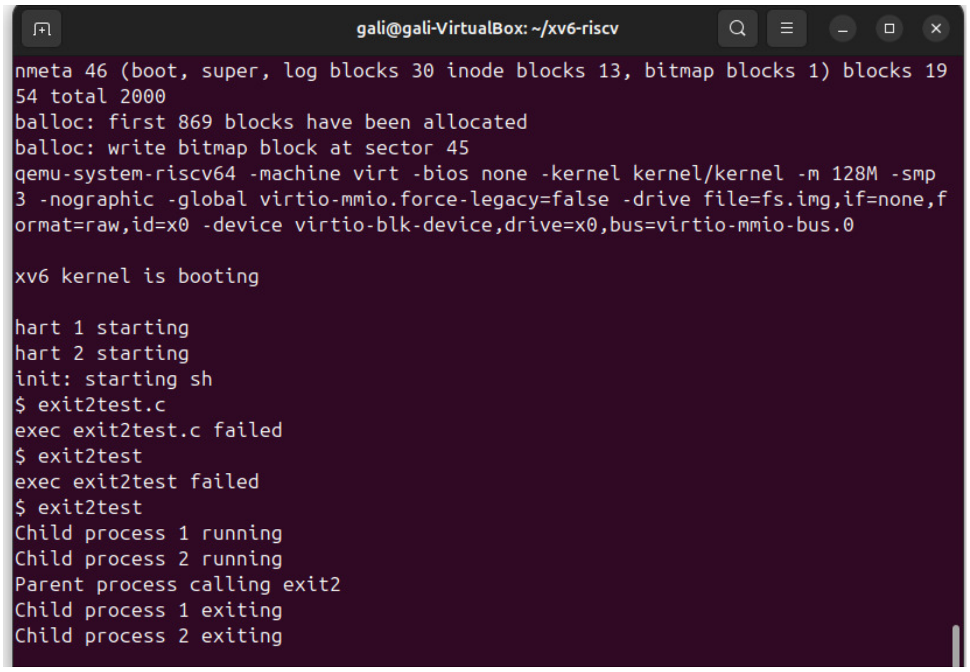
```
riscv64-linux-gnu-ld -z max-page-size=4096 -T user/user.ld -o user/_pause_test u
ser/pause_test.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_pause_test > user/pause_test.asm
riscv64-linux-gnu-objdump -t user/_pause_test | sed '1,/SYMBOL TABLE/d; s/ .* /
/; /^$/d' > user/pause_test.sym
mkfs/mkfs fs.img README user/_cat user/_echo user/_forktest user/_grep user/_ini
t user/_kill user/_ln user/_ls user/_mkdir user/_rm user/_sh user/_stressfs user
/_usertests user/_grind user/_wc user/_zombie user/_pause_test
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 19
54 total 2000
balloc: first 800 blocks have been allocated
balloc: write bitmap block at sector 45
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp
3 -nographic -global virtio-mmio.force-legacy=false -drive file=fs.img,if=none,f
ormat=raw,id=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$ pause_test
Running pause-resume demo.
Running...
```

Figure 2: Another Output of S

4.2 Exit2()



gali@gali-VirtualBox: ~/xv6-riscv

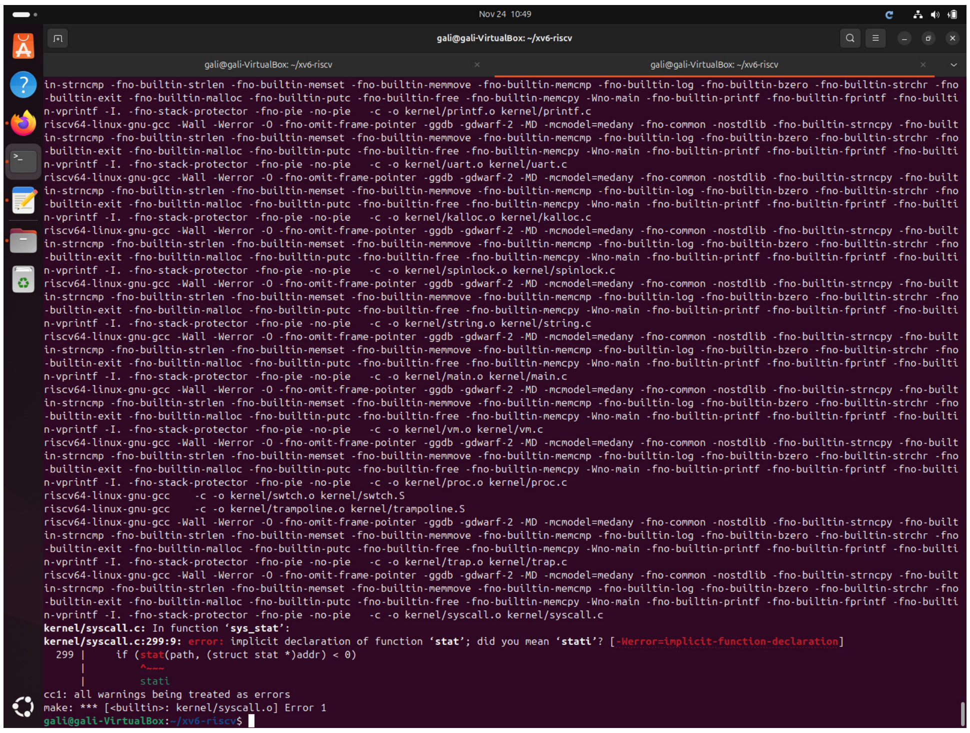
```
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 19
54 total 2000
balloc: first 869 blocks have been allocated
balloc: write bitmap block at sector 45
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp
3 -nographic -global virtio-mmio.force-legacy=false -drive file=fs.img,if=none,f
ormat=raw,id=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$ exit2test.c
exec exit2test.c failed
$ exit2test
exec exit2test failed
$ exit2test
Child process 1 running
Child process 2 running
Parent process calling exit2
Child process 1 exiting
Child process 2 exiting
```

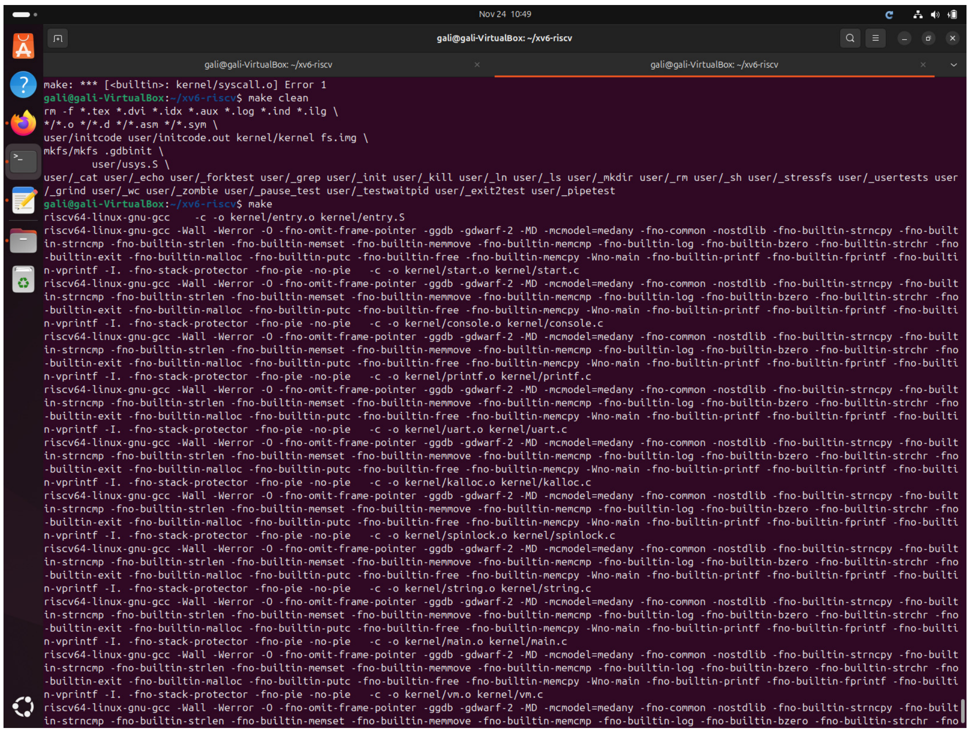
Figure 3: Output of exit2() system call

4.3 pipes


Nov 24 10:49
gali@gali-VirtualBox: ~/xv6-riscv
gali@gali-VirtualBox: ~/xv6-riscv
gali@gali-VirtualBox: ~/xv6-riscv

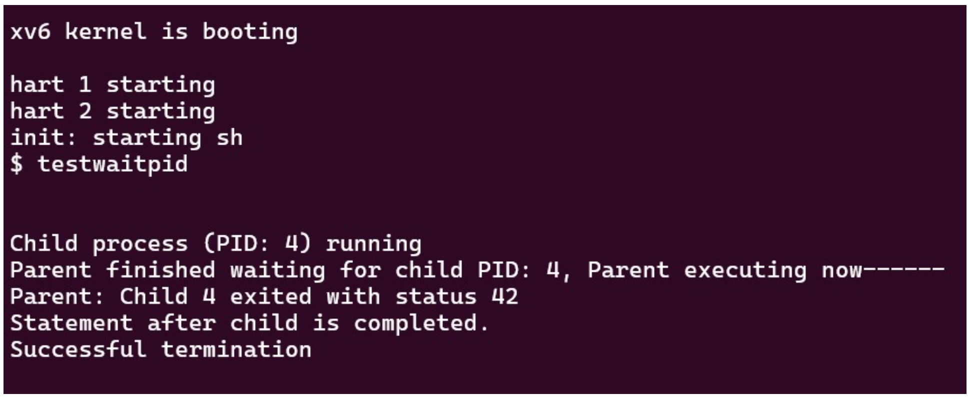
```
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/printf.o kernel/printf.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/uart.o kernel/uart.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/kalloc.o kernel/kalloc.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/spinlock.o kernel/spinlock.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/string.o kernel/string.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/main.o kernel/main.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/vm.o kernel/vm.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/proc.o kernel/proc.c
riscv64-linux-gnu-gcc    -c -o kernel/swtch.o kernel/swtch.S
riscv64-linux-gnu-gcc    -c -o kernel/trampoline.o kernel/trampoline.S
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/trap.o kernel/trap.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/syscall.o kernel/syscall.c
kernel/syscall.c: In function ‘sys_stat’:
kernel/syscall.c:299:9: error: implicit declaration of function ‘stat’; did you mean ‘stati’? [-Werror=implicit-function-declaration]
  299 |     if (stat(path, (struct stat *)addr) < 0)
      |         ^~~~
      |         stati
cc1: all warnings being treated as errors
make: *** [<builtin>: kernel/syscall.o] Error 1
gali@gali-VirtualBox:~/xv6-riscv$ 
```

Figure 4: Output of Pipes


Nov 24 10:49
gali@gali-VirtualBox: ~/xv6-riscv
gali@gali-VirtualBox: ~/xv6-riscv
gali@gali-VirtualBox: ~/xv6-riscv

```
make: *** [<builtin>: kernel/syscall.o] Error 1
gali@gali-VirtualBox:~/xv6-riscv$ make clean
rm -f *.tex *.dvi *.idx *.aux *.log *.ind *.ilg \
*/*.o */*.d */*.asm */*.sym \
user/initcode user/initcode.out kernel/kernel fs.img \
mkfs/mkfs .gdbinit \
        user/usys.S \
user/_cat user/_echo user/_forktest user/_grep user/_init user/_kill user/_ln user/_ls user/_mkdir user/_rm user/_sh user/_stressfs user/_usertests user
/_grind user/_wc user/_zombie user/_pause_test user/_testwaitpid user/_exit2test user/_pipetest
gali@gali-VirtualBox:~/xv6-riscv$ make
riscv64-linux-gnu-gcc    -c -o kernel/entry.o kernel/entry.S
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/start.o kernel/start.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/console.o kernel/console.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/printf.o kernel/printf.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/uart.o kernel/uart.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/kalloc.o kernel/kalloc.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/spinlock.o kernel/spinlock.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/string.o kernel/string.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/main.o kernel/main.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
-builtin-exit -fno-builtin-malloc -fno-builtin-putc -fno-builtin-free -fno-builtin-memcpy -Wno-main -fno-builtin-printf -fno-builtin-fprintf -fno-builti
n-vprintf -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/vm.o kernel/vm.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -fno-common -nostdlib -fno-builtin-strncpy -fno-built
in-strncmp -fno-builtin-strlen -fno-builtin-memset -fno-builtin-memmove -fno-builtin-memcmp -fno-builtin-log -fno-builtin-bzero -fno-builtin-strchr -fno
```

Figure 5: Output of Pipes



```
xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$ testwaitpid


Child process (PID: 4) running
Parent finished waiting for child PID: 4, Parent executing now------
Parent: Child 4 exited with status 42
Statement after child is completed.
Successful termination
```

Figure 6: Output of Wait pid

5 Conclusion

This project focused on extending the functionality of the xv6 operating system by modifying and introducing new system calls. Key areas such as process creation, inter-process communication (IPC), signal handling, and synchronization mechanisms were enhanced to improve system capability. The modifications were thoroughly tested using custom user programs, demonstrating their practical applicability. This work highlights the intricacies of kernel-level programming and the importance of efficient system call management. Overall, the project contributes to a deeper understanding of operating system design and functionality.