

Lab Assignment 2: Hands-on with xv6 OS

Task-1.1: Write user-level sleep program for xv6

CODE:-

```
#include "kernel/types.h"
#include <kernel/stat.h>
#include <user/user.h>

int main(int argc, char *argv[])
{

    int tick;
    if(argc<2)
    {
        printf("No sufficient argument\n");
        exit(-1);
    }

    tick = atoi(argv[1]);
    printf("Time for sleep is %d ",tick);
    sleep(tick);
    exit(0);

    return 0;
}
```

Explanation:-

First I am including all the necessary library of the xv6 os as previous library like #include <stdio.h> and others are not recognised.

Then in main function as an argument passing argc (argument count and argument vector to take input from the command line)

Then I have checked if the argument count is 1 , it means no sufficient argument has been given.

And used “atoi” function to convert into integer as argument is character and then after used system call “sleep” and given argument tick to it

After execution exiting()

Screenshot of output:

```
(base) ash@ash-Inspiron-15-3511:~$ cd xv6-riscv
(base) ash@ash-Inspiron-15-3511:~/xv6-riscv$ make qemu
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
$ sleep-CE21BTECH11008 10
Time for sleep is 10 $
```

Task-1.2: Write pingpong for IPC between two processes for xv6

CODE:

```
#include "kernel/types.h"
#include <kernel/stat.h>
#include <user/user.h>

int main() {
    // for each direction one pipe
    int pipe1[2], pipe2[2];

    // Create pipes
    if (pipe(pipe1) == -1 || pipe(pipe2) == -1) {
        fprintf(2, "Error: Failed to create pipes\n");
        exit(1);
    }

    // child process
```

```

int pid = fork();

if (pid == 0) {

    close(pipe1[1]); // write end for the child process
    close(pipe2[0]); // read end for the child process
    char read_child[1]; // character to read
    read(pipe1[0], read_child, 1);
    printf("%d: recieved ping\n" , getpid());
    write(pipe2[1], "pong", 1);
    close(pipe1[0]);
    close(pipe2[1]);
    exit(0);
} else {

    close(pipe1[0]); // read end for the parent process
    close(pipe2[1]); // write end for the parent process
    char read_parent[1]; // character to read
    write(pipe1[1], "ping", 1);
    read(pipe2[0], read_parent, 1);
    printf("%d: recieved pong\n" , getpid());
    wait(0);
    close(pipe1[1]);
    close(pipe2[0]);
    exit(0);
}

return 0;
}

```

Explanation:-

At first added library similar to previous and then created two pipe using “pipe()” system call and passed two array name pipe1 and pipe2 as each process will read and write both thus for given pipe one end will be read end and other will be write end. After fork, both parent and child have file descriptors referring to the pipe. The child calls close to make file descriptor zero refer to the read end of the pipe, closes the file descriptors. Both the parent and child closes the read side of the pipe, writes to the pipe, and then closes the write side. If no data is available, a read on a pipe waits for either data to be written or for all file descriptors referring to the write end to be closed; in the latter case, read will return 0. Also I am printing ping and pong after they are reading. And at last I am exiting with exit(0) system call with return value of 0.

OUTPUT:

```
TECH11008.asm
riscv64-linux-gnu-objdump -t user/_pingpong-CE21BTECH11008 | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/pingpong-CE21BTECH11008.sym
mkfs/mkfs fs.img README 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/_sleep-CE21BTECH11008 user/_pingpong-CE21BTECH11008
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 1954 total 2000
balloc: first 810 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,format=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
$ pingpong-CE21BTECH11008
4: recieved ping
3: recieved pong
$
```