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.

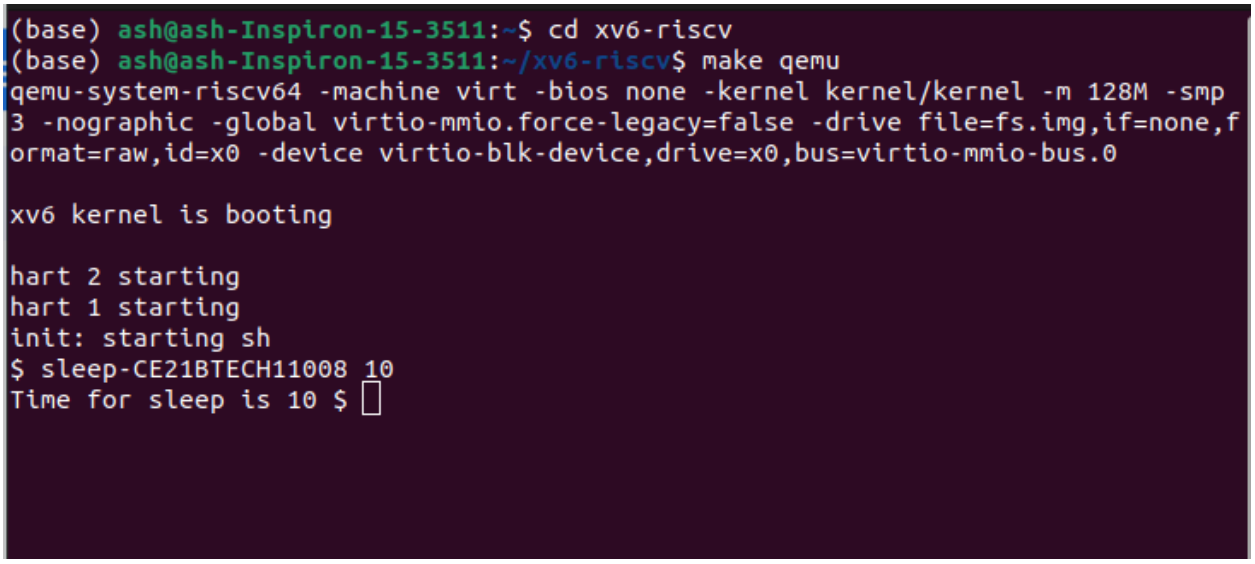
Then in main function as an argument passing argc (argument count and argument vector to take input from the command line)

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:



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

CODE:

```c
#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.Bothe 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 descrip-tors 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: