Week13 Assignment

Please complete a report in English in English in English in English and upload the corresponding codes.

The files should be uploaded directly without compression without compression without compression without compression

The files to be submitted for this assignment are:

1. report.pdf
2. Week13.zip

1. I/O [30 pts]

Please read "Three Easy Pieces" Ch36 https://pages.cs.wisc.edu/~remzi/OSTEP/file-devices.pdf, and answer the following questions:

(1) What are the pros and cons of polling and interrupt-based I/O?

(2) What are the differences between PIO and DMA?

(3) How to protect memory-mapped I/O and explicit I/O instructions from being abused by malicious user process?

2. Condition variable [30 pts]

Please implement the condition variable in ucore by the already implemented wait queue or semaphore.

Requirements:

1. Please complete the definition of the condition variable in `condvar.h`
2. Please implement the related functions of condition variables in `condvar.c`
3. We have used these functions in check_milk, please make sure your implementation can make check_milk run in valid order. (Please release annotations of check_milk in init_main in proc.c for testing, and annotate check_sync).

Please include your design idea, code(screen-shot) and the running result(screen-shot) in your report.

3. Bike [40 pts]

Use **semaphore and condition variable** implemented in Exercise 1 to solve the following synchronous mutual exclusion problem.

There is a bicycle production line, and there are 3 workers whose activities are:

```
//worker1
```

```
while(1){
    Make a bike rack
}

//worker2
while(1){
    Produce two wheels
}

//worker3
while(1){
    Get a bike rack and two wheels and assemble the bike
}
```

There are some rules:

- Only one worker is working at the same time
- The production process must follow: worker1=>worker2=>worker3
- Product bikes continuously

Please complete the corresponding implementation in check_exercise.

Note the use of the do_sleep function.

Please include your design idea, code(screen-shot) and the running result(screen-shot) in your report.

Please add the following content in function `init_mian` in `proc.c`

```
//extern void check_milk(void);
//check_milk();

extern void check_exercise(void);
check_exercise();

//---already have--------
while (do_wait(0, NULL) == 0) {
schedule();
}
```

Add a file `check_exercise.c` in directory `kern/sync`

```
// kern/sync/check_exercise.c
#include <stdio.h>
#include <proc.h>
#include <sem.h>
#include <assert.h>
#include <condvar.h>

struct proc_struct *pworker1,*pworker2,*pworker3;


void worker1(int i)
{
    while (1)
    {
```

```
        cprintf("make a bike rack\n");

    }
}


void worker2(int i)
{
    while (1)
    {
        cprintf("make a front wheel\n");

        cprintf("make a rear wheel\n");
    }
}




void worker3(int i){
    while (1)
    {
        cprintf("assemble a bike\n");
    }
}


void check_exercise(void){

    //initial


    int pids[3];
    int i =0;
    pids[0]=kernel_thread(worker1, (void *)i, 0);
    pids[1]=kernel_thread(worker2, (void *)i, 0);
    pids[2]=kernel_thread(worker3, (void *)i, 0);
    pworker1 = find_proc(pids[0]);
    set_proc_name(pworker1, "worker1");
    pworker2 = find_proc(pids[1]);
    set_proc_name(pworker2, "worker2");
    pworker3 = find_proc(pids[2]);
    set_proc_name(pworker3, "worker3");
}
```