

CS302:

Assignment8 Report

Name: 陆菡芸 SID: 12011537

1 Answer of Question 1

(1) Pros and cons of polling and interrupt-based I/O

- Polling:

Pros: This basic protocol works simple by just asking what is going on once a while.

Cons: Inefficient and inconvenient. Since polling executes until the device finished its I/O, it wastes a great deal of CPU time just waiting for the potentially slow device.

- Interrupt-based:

Pros: It saves CPU time since when a process is operating with I/O it will be put to sleep and context switch to another process. The use of disk is also utilized during the middle stretch of time and can inform the OS to wake up once after the disk request is finished.

Cons: If a device performs quickly and polling can immediately finds its completion, interrupt-based still needs to perform context switch, interrupt handling which is time consuming and slows down the system.

(2) Differences between PIO and DMA

PIO and DMA are both methods helping data to move between memory and disk. PIO involves CPU to move data. But DMA bypass CPU and the OS program the DMA engine to transfer the data. A DMA engine is a very specific device that can organize transfers between devices and main memory without much CPU intervention.

(3) Memory-mapped I/O and explicit I/O instructions protection

1. Ensure these instructions are privileged, the OS should be the only entity allowed to directly communicate with them.

2 Answer of Question 2

The condition variable structure has two members: one semaphore and one counter value to record the waiting threads. At the very beginning, initialize the condition variable by initialize its semaphore and set the counter to 0.

For the `cond_signal` function, it is called when the waiting is over and mom or sister needs to buy milk to fill the fridge. Therefore, if the waiting threads are larger than 0, then wake up and unlock one thread in the waiting queue of the semaphore.

For the `cond_wait` function, it releases lock, put thread to sleep until condition is signaled. When thread wakes up again, it will reacquire the lock.

The following codes are the detailed implementations.

```
typedef struct condvar
{
    //=====your code=====
    semaphore_t sem;
    int count;
} condvar_t;
```

(a) Struct

```
void cond_init(condvar_t *cvp)
{
    //=====your code=====
    sem_init(&cvp->sem, 0);
}
```

(b) Init function

```
// Unlock one of threads waiting on the condition variable
void cond_signal(condvar_t *cvp)
{
    //=====your code=====
    if (cvp->count > 0)
    {
        up(&cvp->sem);
        cvp->count--;
    }
}
```

(c) Signal function

```
void cond_wait(condvar_t *cvp, semaphore_t *mutex)
{
    //=====your code=====
    // release mutex and sleep on cvp
    cvp->count++;
    // cprintf("%s", "fal?");
    up(mutex);
    down(&cvp->sem);
    down(mutex);
}
```

(d) Wait function

Fig. 1: Condition Variable Implementation

The results are shown below. Me and dad first consumed 100 milk by eating 20 a time. When mom and sister checked the fridge in this process, there were milk left so they went to wait until the next time dad signaled them to buy the milk.

<pre>memory management: default_pmm_manager physcial memory map: memory: 0x08800000, [0x80200000, 0x885ffff]. sched class: stride scheduler SWAP: manager = fifo swap manager ++ setup timer interrupts you check the fridge. you eating 20 milk. sis checks the fridge. sis waiting. Mom checks the fridge. Mom waiting. Dad checks the fridge. Dad eating 20 milk. Dad checks the fridge. Dad eating 20 milk. Dad checks the fridge. Dad eating 20 milk. you check the fridge. you eating 20 milk. you check the fridge. you eating 20 milk. Dad checks the fridge. Dad tell mom and sis to buy milk sis goes to buy milk... sis comes back. sis puts milk in fridge and leaves. sis checks the fridge. sis waiting.</pre>	<pre>Dad checks the fridge. Dad eating 20 milk. you check the fridge. you eating 20 milk. you check the fridge. you eating 20 milk. Dad checks the fridge. Dad eating 20 milk. Dad checks the fridge. Dad eating 20 milk. Dad checks the fridge. Dad eating 20 milk. you check the fridge. you tell mom and sis to buy milk Mom goes to buy milk... Mom comes back. Mom puts milk in fridge and leaves. Mom checks the fridge. Mom waiting. you check the fridge. you eating 20 milk. Dad checks the fridge. Dad eating 20 milk. Dad checks the fridge. Dad eating 20 milk. you check the fridge. you eating 20 milk. you check the fridge. you eating 20 milk. Dad checks the fridge.</pre>	<pre>Dad tell mom and sis to buy milk sis goes to buy mlk... sis comes back. sis puts milk in fridge and leaves. sis checks the fridge. sis waiting. Dad checks the fridge. Dad eating 20 milk. you check the fridge. you eating 20 milk. QEMU: Terminated (base) ldy12011537@ludiyun-R0G:~/Desk</pre>
--	--	--

Fig. 2: Final Results