CS502 PROJECT1

Chengjiao Yang 10/14/2013

Architectural And Policy Overview

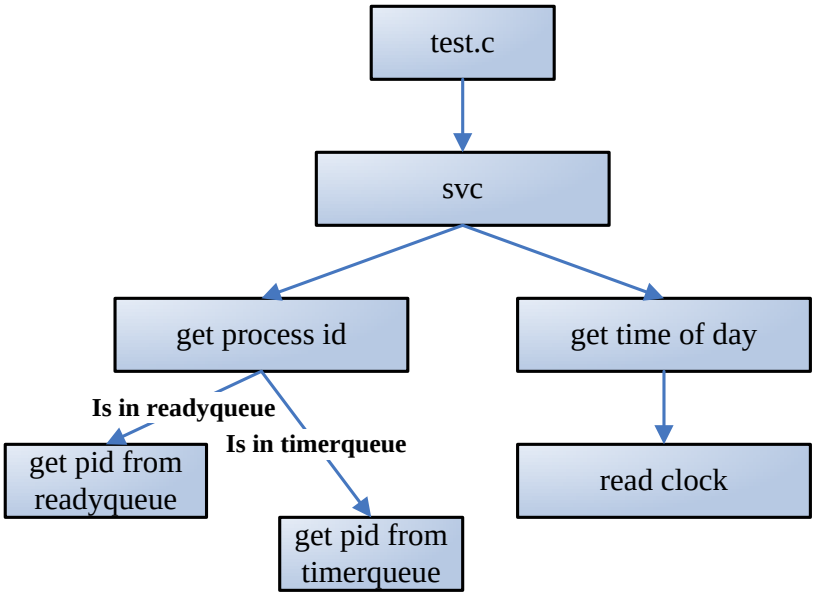
This project contains multi-elements, we get request from user in test.c, then handle it in base.c. According to this framework, we write routines for system process such as OSCreateProcess, dospprint. Queue, message and other routines are also designed for handling different requests from user. Interrupt handler and fault are important part and well-designed in this project too. To make system going well, some suitable mechanism are Applied. For more detail, please see the context below.

WHAT Is Included

1. Communicate with hardware
   Set timer, Fault_handler
2. Queue process
   Add node to queue, get information from queue, delete node of queue, FIFO order, priority scheduler logic.
3. PCB
   Make process, change priority, switch PCB
4. Message process
   Send message, receive message, search message
5. Print out
   dospprint integrate lock and queue display
6. Debug
   ListQueue, ListTimerQueue, ListReadyQueue, ListSuspendQueue, ListTwoQueue
7. Process Synchronization
   Lock of queue and messagelist to prevent the conflict

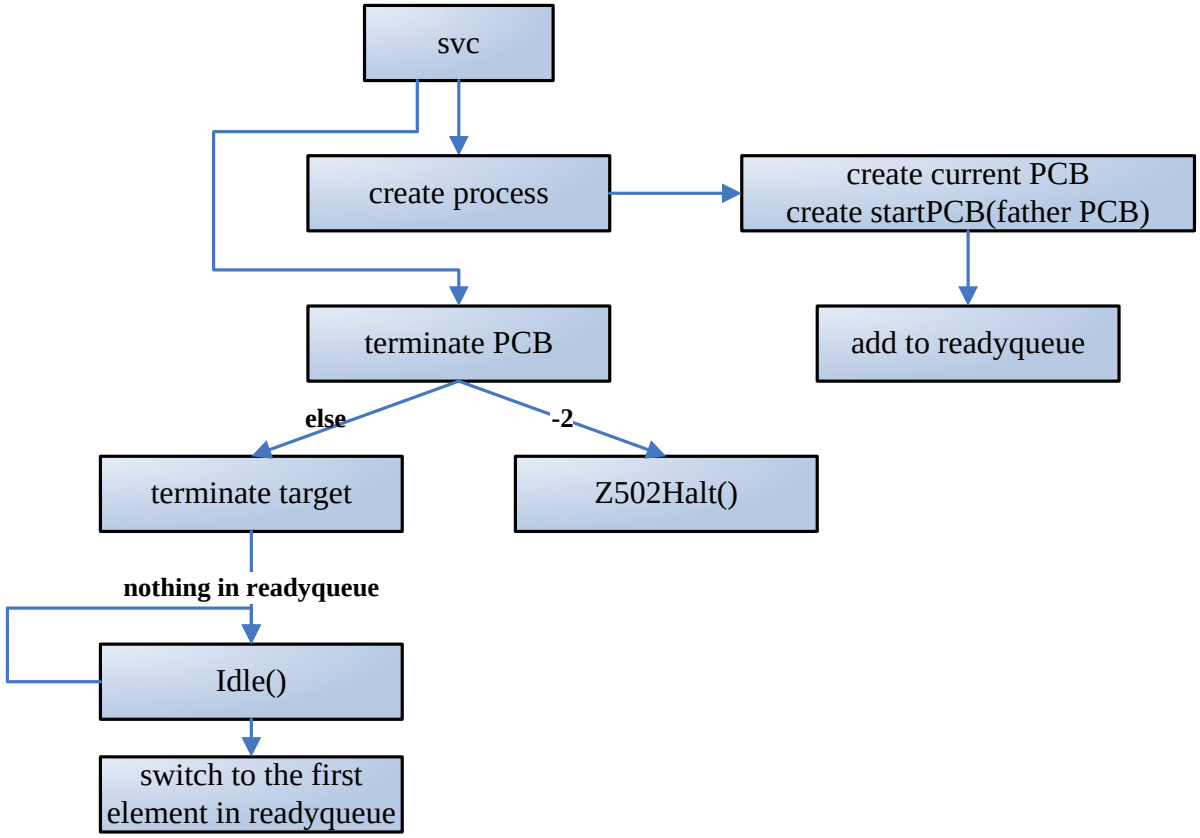
High Level Design

I tried to put everything in one graph, but then I found it's impossible to make a such big diagram and clearly show the feather and organization of my system. Thus I divide it into six indivisible parts, every part is connected by svc.



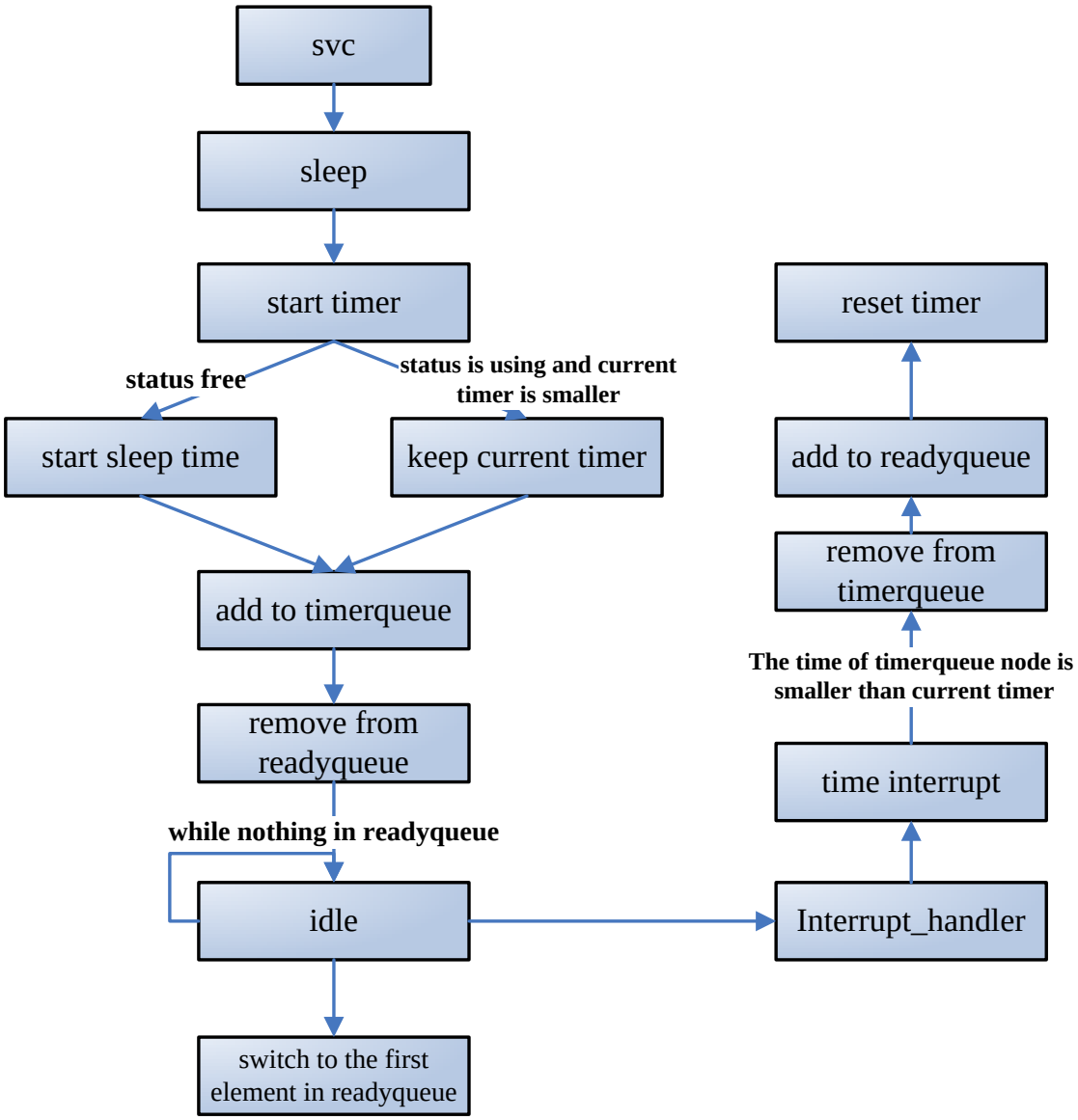
Justification: get processed, get time of day

This diagram show how the user and OS link together, test.c send system call to OS, the SVC get the request and dispatch the request to matched routines. Here we show the get time of day and get processed, the first one read the z502clockstatus and return the time, the second one judge whether the PID is in readyqueue and timerqueue, then get PID in queue and return.



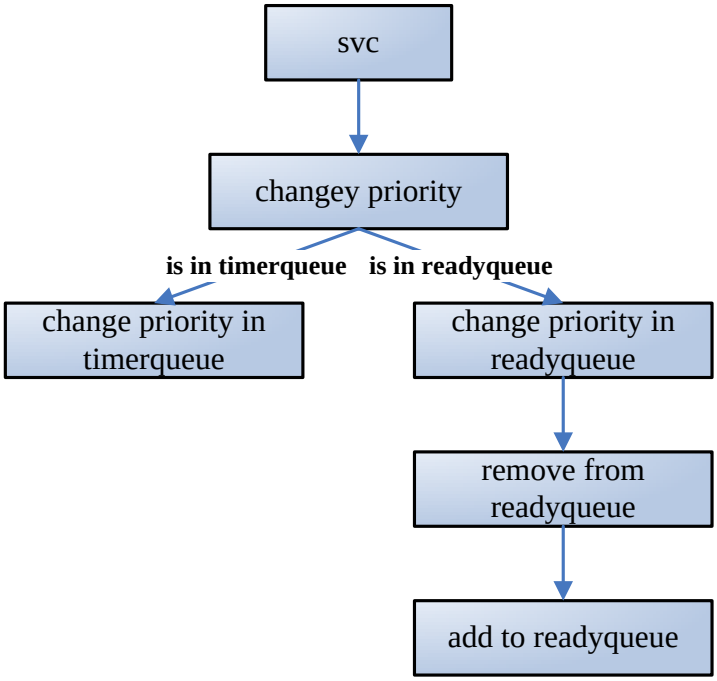
Justification: create process, terminate process

After getting create process request from user, there is a routine called OSCreateProcess handle it, it first create current PCB and startPCB(this is father PCB), then add the PCB to readyqueue. When the OS get terminate request, OS will terminate process according to the PID received, remove the matched PCB from readyqueue, if nothing left in readyqueue, it loops until the PCB appears in readyqueue and switch to the first one in readyqueue.



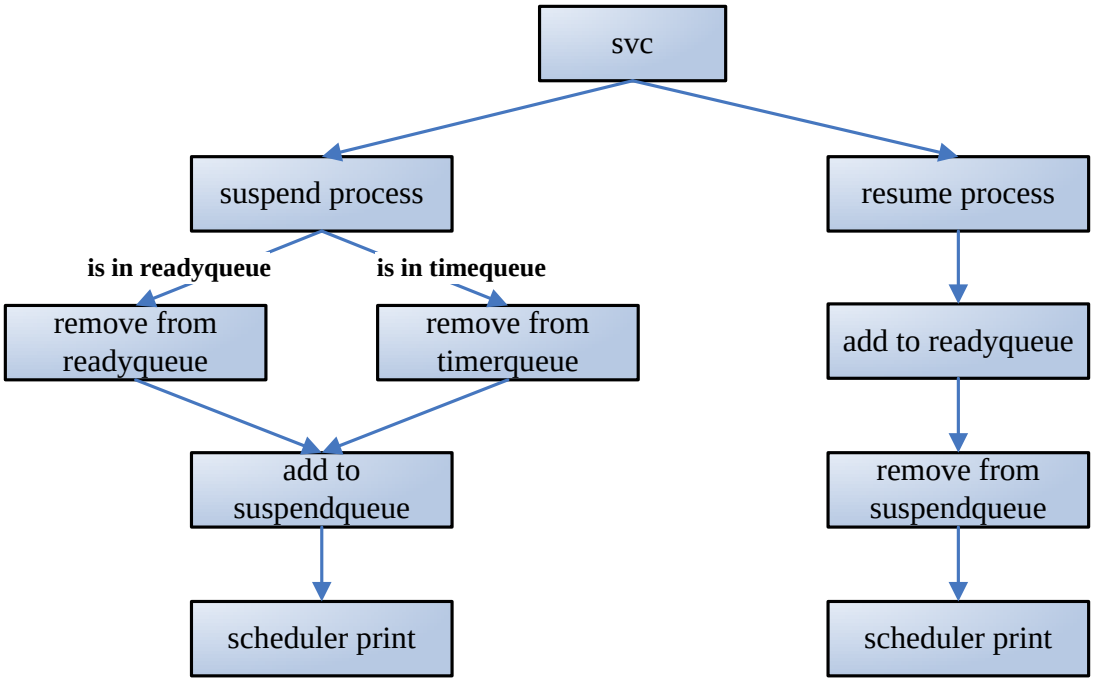
Justification: sleep, interrupt

Sleep and interrupt are always hang out together, when OS gets sleep request, it set timer according to the timer status, then add PCB to timerqueue, remove PCB from readyqueue, when nothing left in readyqueue, loops until something appear and switch. The interrupt(now we only handle with time interrupt) does the things just opposite to sleep.



Justification: change priority

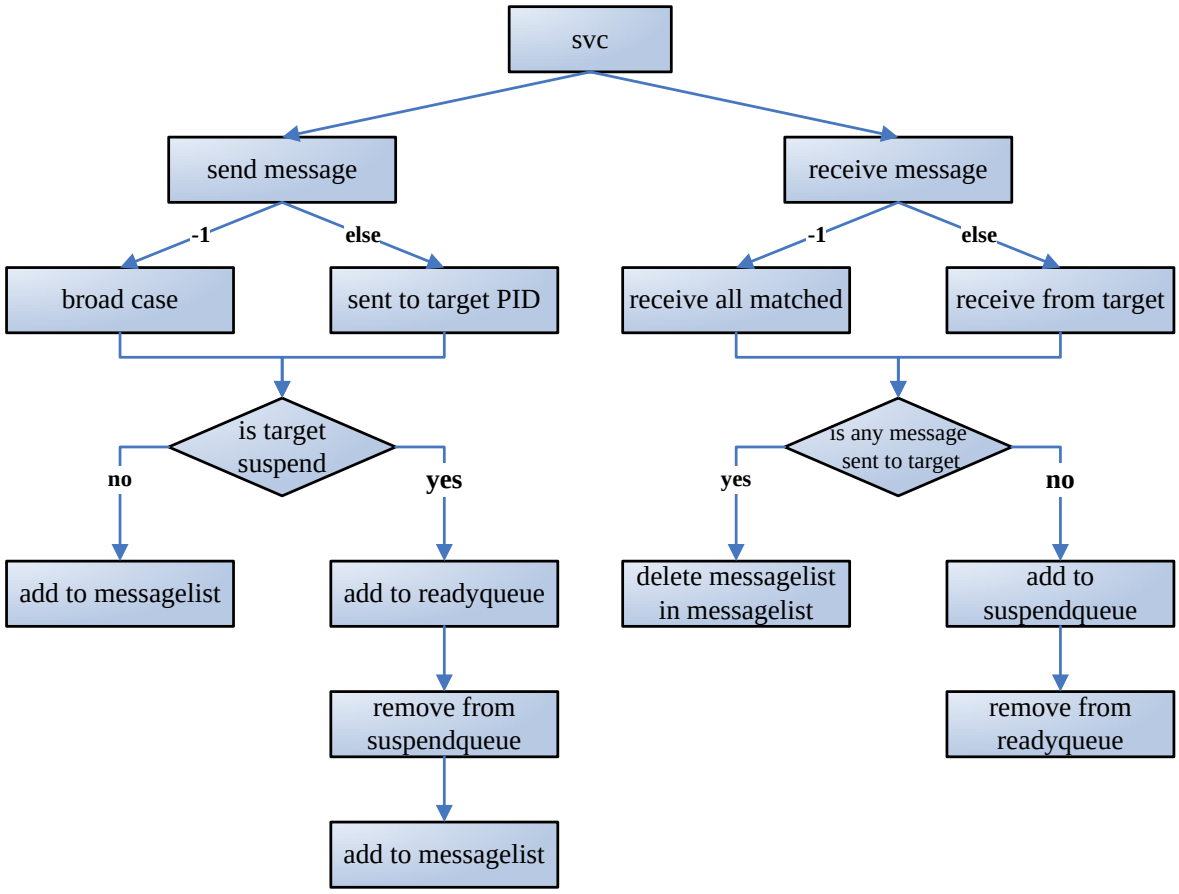
Change priority is actually logic focus on the dispatch scheduler. If no priority scheduler was using, then it's FIFO scheduler. In project1 we use priority scheduler, mean the queue is ordered by priority, but that is different between readyqueue and timerqueue, since we don't case about the priority in timerqueue, so my logic is change priority attribute in timerqueue, but re-insert PCB by priority in readyqueue.



Justification: suspend process, resume process, scheduler print

Literally, suspend process and resume process should do the opposite things, in fact

they did. In my logic, the only difference between them is when doing suspend, we have to judge weather the process is in readyqueue or in timerqueue, when doing resume, then just put back to readyqueue. In addition, I show how scheduler print work in this diagram, I always print the result after operation.



Justification: suspend message, receive message

This part is interesting, send and receive message, image it likes msn stuff. However, it is just array things. When send message, resume the target process, add the message to messagelist(a structure array I design). When receive message, if there is matched message, then delete that message it messagelist, if no message recevice, suspend the current PCB. Here is a trick, because everything the test1j_echo will receive message first, but we know receive lead to suspend, suspend lead to switch context, to keep the system robust, there is loop contain receive and suspend, mean every time the process will receive message no matter how they switch.

Additional Features

In test1i and test1j, we are doing the send message and receive message simulation. That's an interesting function, however, we just add message to and delete message from the message list during test. I think we could make this the message more

powerful, so I designed the test1m. In test1m, I allow the process to send message to change the priority and response to the “say hello” command. To achieve this function, I add a messageprocess() routine in receive_message() svc. When I PCB receive a message from father PCB that require child PCB to say hello 20 times, the child should response, I think this quite make sense.

Anomalies or bugs

1.Scheduler print can only show 4 number of time, that’s no convince.
2.Kill mode will lead to unknown error
3.Some code are ok in vs, but not ok in gcc. For example:

```
char *dd;
dd = “hahaha”;
```

is ok in vs, but it doesn’t work in gcc.
4.We don’t have a routine can get current context in base.c, we have to write ourself.