Assignment 6

Group Number - 2

Group Members

---

1. Ashrujit Ghoshal (14CS10060)
2. Sayan Ghosh (14CS10061)

==========

Files added :

1. Signal.c
2. Signal.h

Files modified :

1. Thread.c
2. Thread.h
3. Makefile.build

Description of Signal.c

==========

Data Structures :

Signal structure consists of the following :

1. enum signum
2. tid_t sender
3. tid_t receiver
4. A list element named sigelem to insert signal into the doubly linked list of signals

The signum is of enum type and can be one of the five values SIG_CHLD, SIG_CPU, SIG_UNBLOCK, SIG_USER, SIG_KILL

There is an enum for specifying the signal flag like SIG_DFL and SIG_IGN.
SIG_DFL makes a call to the default handler for that corresponding signal, whereas SIG_IGN is used to ignore the signal.

The function sigprocmask also takes 3 types of flags to denote the type of the action needed to be done. These flags are SIG_BLOCK, SIG_UNBLCK, SIG_SETMASK.

Important functions and their descriptions :

1. sighandler_SIG_KILL_DFL()
   This is the default signal handler for the SIG_KILL signal. It calls a helper function which in turn invokes the thread_exit() function to kill the thread.

2. sighandler_SIG_CPU_DFL()
   This is the default signal handler for the SIG_CPU signal. It calls a helper function which in turn invokes the thread_exit() function to kill the thread. It prints a message to address the fact that the action for the signal has been taken.

3. sighandler_SIG_CHLD_DFL()
   This is the default signal handler for SIG_CHLD signal. It decrements the number of children of the parent thread and prints a message.

4. sighandler _SIG_USER_DFL()
   This is the default handler for the SIG_USER signal. It prints a message specifying the sender and recipient of the signal.

5. sighandler_SIG_UNBLOCK_DFL()
   This is the default handler for the SIG_UNBLOCK signal. If the status of the thread is blocked then the thread_unblock() function is called and the recipient thread is passed as an argument to the thread_unblock function. The thread_unblock function is a predefined function present in pintos codebase which changes the status of the thread, unblocks and puts it back in the ready queue for scheduling.

6. setlifetime(int x)
   It sets the lifetime of the thread. This function is called inside a thread and it fixes the lifetime.If this is not invoked thread has unlimited lifetime.

7. signal(enum signumber signum,enum sig_flags flag)
   This is used to specify a flag for a signal. SIG_DFL flag denotes to call the default signal handler when the signal is received, whereas the SIG_IGN flag tells to ignore the signal.

8. kill(enum signumber signum,tid_t rec)
   This function is used to send a signal from one thread to another. The first argument is the thread id and the second is the signum. The signal is registered and put into a list and when the recipient thread is next scheduled then the signals are delivered.

9. sigemptyset()
   It creates an empty mask set. So all the signal are unmasked.

10. sigfullset()
    It creates a fully masked set. In our case sigmask is a 4-bit number. So it basically sets the sigmask to 15 (1111).

11. sigdelset()
    It removes a signal from the mask set. On the basis of the signum the particular bit is set to zero and the corresponding decimal number (int) is stored in the sigmask field of the thread.

12. sigaddset()
    It adds a signal to the mask set. On the basis of the signum the particular bit is set to 1 and the corresponding decimal number (int) is stored in the sigmask field of the thread.

13. sigprocmask()
    It is used to set a mask, block or unblock signals for a thread. It takes as argument -- a flag denoting the type of action to be performed, a new set (sigmask) and an old set (old sigmask).
    If the flag is SET_MASK then the sigmask of the thread is set to the new set.
    If the flag is SIG_BLOCK then a bitwise OR is taken between the original sigmask of the thread and the new set and the result is set as the sigmask.
    If the flag is SIG_UNBLCK then bitwise AND is taken between the original sigmask and the complement of the new set and the result is set as the sigmask.

Changes made in thread.h
==========

Change in the thread data structure :
Some fields have been added --

1. Numchild
2. Totalchild
3. Unblock_delivered
4. Tick_count
5. isCPULimitOver
6. Max_ttl
7. create _timestamp

Numchild keeps track of the number of children of the thread alive and totalchild keeps track of the number of children created.
Tick_count keeps track of the number of clock ticks the process has spent (either in execution or in waiting)
isCPULimitOver is a flag to denote if the CPU lifetime is over.
Max_ttl is the maximum time a process can live.
Create_timestamp stores the time when the thread is created.

Modifications in thread.c
==========

At the end of the schedule() function we call handle_all_signals() function to check the signals to be delivered to the thread which has been scheduled.
In handle_all_signals() we traverse the list of signals (list of signal_book structure). Here the other_list is traversed in which we check for all the signals except SIG_UNBLOCK. For SIG_CPU we check if the flag isCPULimitOver is set.We deliver the queued signals whose receiver is the next thread to run. We also traverse the unblock_list and unblock all the blocked threads in the list.

In thread_tick function we check if a thread has exceeded its lifetime. We traverse list of all threads and set isCPULimitOver to 1 if a thread has exceeded its lifetime.

Modifications in Makefile.build
==========
We added the following line to compile and link signal.c in our kernel:
threads_SRC += threads/signal.c

Discussions regarding the design choice
==========

We take two global list of signals, one for SIG_UNBLOCK and one for the others(except SIG_CPU).
We did not take a separate list of signals inside each and every thread because then while registering the signal one thread needs to access the address space of another thread which is not desired.
During insertion into the list we check if for the same receiver a signal with the same signum exists or not. In case it exists, we update the sender tid and replace it with the tid of the latest thread which signals. However this is not done in case the signum is SIG_CHLD as we want to

keep track of all the dead children threads. This is in accordance with the requirements of the assignment.