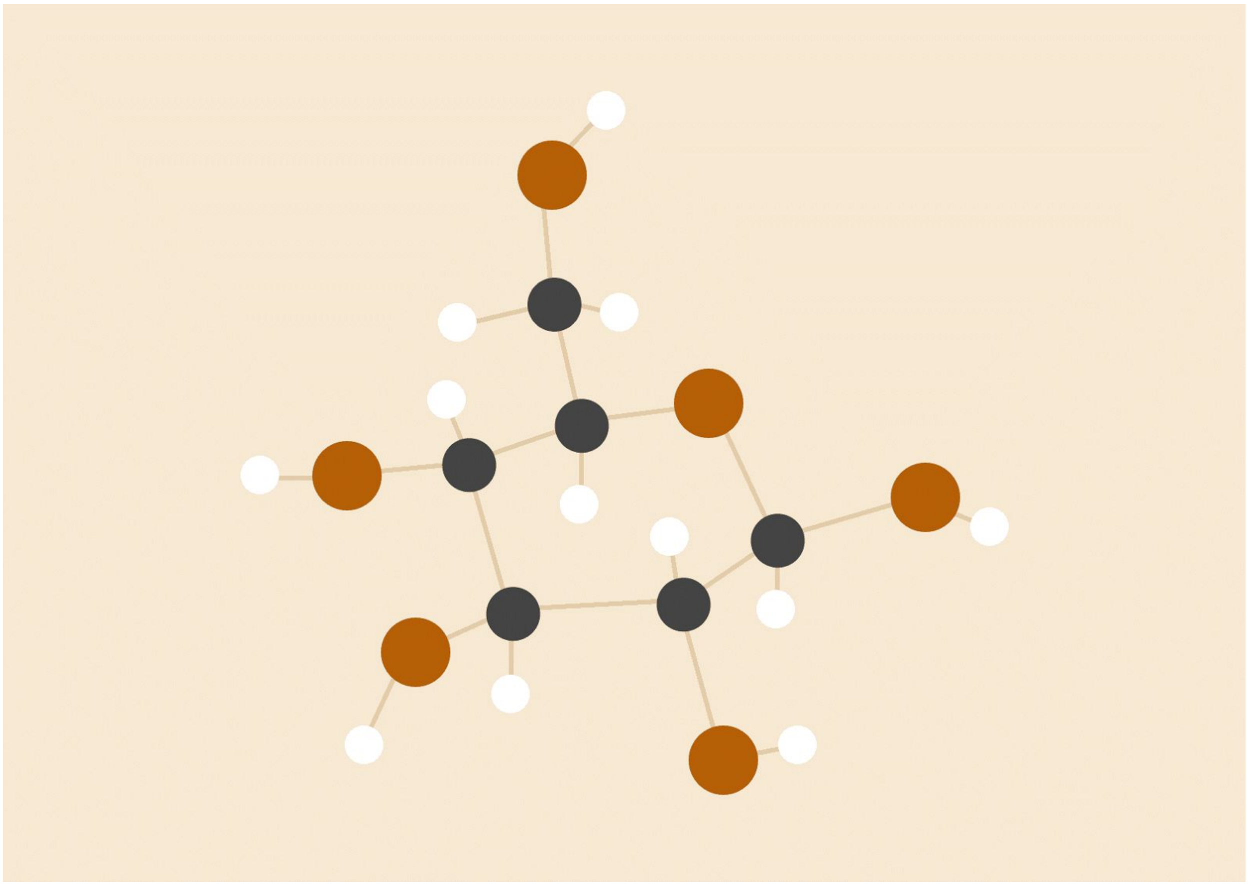
OPERATING SYSTEMS

Implementing Signals for PINTOS



Arijit Panigrahy | 14CS30005

Aniket Choudhary | 14CS30004

13.04.2017

Operating Systems Laboratory

GROUP:- 1

OBJECTIVE

To support signals as well it's handlers in pintos. Also support signal masking in pintos.

FILES MODIFIED

1) Signal.c (Created)
2) Signal.h (Created)
3) Thread.c (Modified)
4) Thread.h (Modified)

DATA STRUCTURES ADDED/CHANGED

1) Thread.h
   Struct thread:-
   parent_tid (parent thread id)
   child_total (total number of children of a thread)
   child_alive (number of alive children)
   Lifetime (MAXIMUM Lifetime of a thread)
   Ticks (Number of time lived by the thread)
   Signals_queue (List of signals received by thread)
   Mask (Information about blocked and unblocked signals)
   Block_elem (If it has to be unblocked)
   Signal[5] (To keep track of sender and recent signal)
2) Signal.h
   Signal_data:-
   signum (SIG_KILL/SIG_USR/ etc..)
   Sender (Sender)
   threadelem (for signal's queue)

FUNCTIONS ADDED/MODIFIED

1. Signal(int signum, handler)
   If signum==SIG_KILL, return 0. If handler=SIG_IGN, change the mask of the thread to block the signal else, change it to unblock the signal.

2. Kill (int tid, int sig)
   It is valid only for SIG_KILL, SIG_USER & SIG_UNBLOCK. It checks the thread's mask whether it has been blocked. If it is blocked, then the signal is ignored. Else, if the signal is SIG_UNBLOCK, the thread gets unblocked. Else, the signal is queued in the signal_queue after initializing the parameters of the signal structure.
3. sigprocmask(int how, set, oldset)
   If how=0, => Signal is blocked, else if it is 1, then signal is unblocked else if it is 2, the mask is set.
4. sigemptyset(set)
   Unblocks all the signal.
5. sigfillset(set)
   Blocks all the signal
6. sigaddset(set,signum)
   Blocks a specific signal specified by signum.
7. sigdelset(set, signum)
   Unblocks a specific signal specified by signum.
8. handler(sender, signum)
   Handles the signal based on the signum and prints the relevant message after setting the count of children in case of SIG_CHILD and calling thread_exit() in case of SIG_CPU.
9. find_parent(const int tid)
   Finds the parent of the thread "tid" from the hash table.
10. thread_compare()
    Compares the tid of two threads.
11. thread_init() [Modified]
    Initializes the to_unblock list.
12. thread_start() [Modified]
    Initializes the hash table.
13. Updating_cpu
    Checks if the lifetime of the current thread is greater than the specified limit, in which case SIG_CPU is queued.
14. Thread_tick [Modified]
    Increases the ticks of all the threads and if the lifetime of the thread is greater than the specified limit, SIG_CPU is queued.
15. Thread_exit [Modified]

Calls update_child() to send SIG_CHLD to parent.

16. update_child

    Finds the parent thread. If parent is not null, sends the SIG_CHLD to parent.
17. Init_thread [Modified]

    Initializes the new parameters of the struct thread such as lifetime, ticks etc.
18. handle_unblock

    Checks the unblock_list and unblocks all the thread from that list. This is as a result of SIG_UNBLOCK signal to the threads.
19. Handle_signal

    Checks the signal queue and calls the appropriate handlers for each of them and removes them from the queue.
20. Schedule

    Calls handle_unblock and handle_signal functions
21. Setlifetime

    Sets the lifetime of the thread.