Report

Group 49

Implementation of Signal Handling in Pintos

DATA STRUCTURES CHANGED

Following things are added in thread.h

int parent; int total_child_count; int child_count;   int life_time;
int ticks; int signal_register[5]; int signal_queue[5]; int signal_mask;
struct list_elem ubelem; extern struct list all_list;extern struct list unblock_list;

Following things are added in signal.c

struct list all_list; struct list unblock_list; void setlifetime(int time);

Following things are added in signal.h

typedef int tid_t;

Following things are added in signal.c

struct list all_list; struct list unblock_list;

FUNCTIONS MODIFIED

In thread.c file:

1. thread_init():

Initialized unblock_list.

2. schedule():

This functions calls signal handlers by checking masks and other blocked signals accordingly after thread_schedule_tail function i.e.., when context switch changes.

3. init_thread():

This function initializes values of the current thread structure.

4. thread_tick():

This function calls the thread_yield() function after each time slice in the queues and increments ticks at each timer tick.

5. setlifetime():

Sets the maximum life time of a thread from the argument.

6. thread_exit():

Exits the current thread and calls child handler for SIG_CHLD if the parent to that child is present.

In signal.c file:

1.CHLD_handler():

This function updates the count of children which has died so far and prints both the total number of children created by X (till the signal is delivered) and the number of children still alive.

2. KILL_handler():

This function simply terminates the receiving thread and prints a message.

3. CPU_handler():

This function prints the maximum lifetime set and terminates the thread.

4. UNBLOCK_handler():

this function unblocks the receiving thread if it is blocked. If the receiving thread is already unblocked, no action is taken.

5. USER_handler():

This function just prints which thread sent the signal to which thread and returns

6. mysignal():

Does same work as linux signal() and since we do not support user-defined signal handlers, the only options supported for the second argument are SIG_IGN and SIG_DFL with same meaning. SIG_KILL cannot be ignored, so calling signal(SIG_KILL, SIG_IGN) returns success but does nothing.

7. kill():

This function does same work as linux signal() and should be valid only for SIG_KILL, SIG_UNBLOCK, and SIG_USR subject to the conditions stated, all other cases are error cases. Returns 0 on success, -1 on error.

8. sigemptyset():

This function initializes the signal set given by *set* to empty, with all signals excluded from the set.

9. sigfillset():

This function initializes *set* to full, including all signals.

10. sigaddset():

This function adds the signal *signum* from *set*.

11. sigdelset():

This function deletes the signal *signum* from *set*.

12. sigprocmask():

It is used to fetch and/or change the signal mask of the calling thread. The signal mask is the set of signals whose delivery is currently blocked for the caller.

FILES MODIFIED

1.thread.c   2.thread.h   3.signal.c   4.signal.h