Name: Lokesh Kodali Name: Umesh Chanumolu
NUID: 001613100

Project: Kernel threads

Implemented System calls

int clone(void (*funct)(void *), void *stack, void *arg) : Implemented clone() system call, which creates other process with new process ID and shares same mappings as its parent. It takes three arguments like the address of function where the thread should start its execution, the address of the user stack and the argument to the function. It pushes the argument into the user stack and pushes a fake return address so that thread will not return.

int join(void **stack) : join system call takes one argument which can store the address of the user stack. It cleans up the zombie thread if present or it will return -1 if there are no threads. If the thread is not completed execution, then the parent thread will go to sleep and wait for child thread to complete.

int sleep_cond(lock_t *) : This system call will make the thread to sleep on respective conditional variable.

int wake_cond(int) : It will wakeup the thread waiting on a particular conditional variable.

User functions implemented in **thread.c** for supporting basic thread creation and synchronization.

int thread_create(thread_t *p, void (*funct)(void *),void *arg) : It will create the thread which starts execution at function "funct" and the sends the argument "arg" to that function. It will allocate 4KB memory using malloc() call and sends it to the clone system call.

int thread_join( ) : It uses the join() system call to reap away the zombie thread or to wait for the child thread to complete its execution and free the user memory.

void lock_init(lock_t *) : It initializes the lock by setting the variable flag to zero. Lock is received as reference to the function.

void thread_lock(lock_t *) : It will try to set the variable 'flag' in the lock datatype. It will take the hardware support by using "xchg" instruction, which will atomically checks the value of flag and sets it if available. It will spin in while loop till the lock is acquired (till flag becomes zero)

void thread_unlock(lock_t *) : this function will set the flag to zero (unlocking) so that other threads which are waiting on the lock can use it and enter the critical section in the code.

void cond_wait(cond_t *, lock_t *) : This function will release the lock and suspends the thread in cond variable.

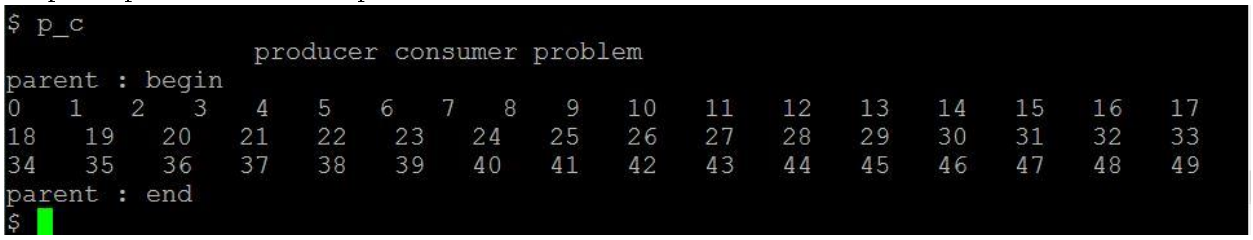
void cond_signal(cond_t *) : It will wake up the threads which are on the cond variable.

void cond_init(cond_t *) : Initializes the queue to store the threads waiting on the "cond" conditional variable.

Modified files:
proc.c, sysproc.c, sysproc.h, usys.S, syscall.c, types.h, user.h, defs.h
Created a new file “thread.c” which contains the thread library (implementation of above user level functions)
Included thread.o in Makefile (in ULIB section)

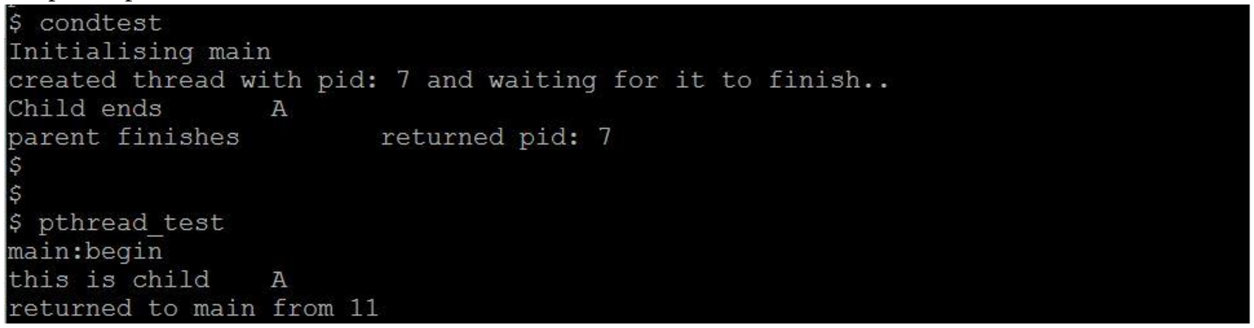
User tests files:
clone_test.c : used to test the implementation of the clone system call. It does not contain join() syscall, so it will try to return to 0xffffffff address and will be killed and will remain as zombie.
pthread_test.c : to test the implementation of thread_create and thread_join functions.
condtest.c : to test the thread synchronization by using locks and conditional variables.

p_c: prodcer consumer problem similar to the code in reference material.
Output of producer consumer problem



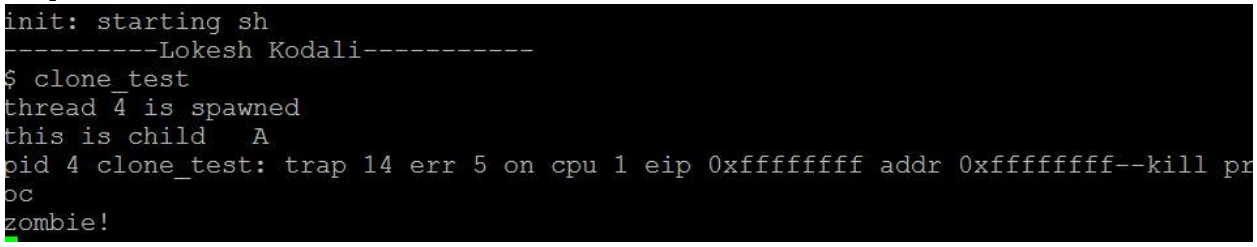
```
$ p_c
                producer consumer problem
parent : begin
0   1   2   3   4   5   6   7   8   9   10   11   12   13   14   15   16   17
18   19   20   21   22   23   24   25   26   27   28   29   30   31   32   33
34   35   36   37   38   39   40   41   42   43   44   45   46   47   48   49
parent : end
$
```

Output of pthread_test.c and condtest.c



```
$ condtest
Initialising main
created thread with pid: 7 and waiting for it to finish..
Child ends       A
parent finishes          returned pid: 7
$
$
$ pthread_test
main:begin
this is child    A
returned to main from 11
```

Output of clone_test.c



```
init: starting sh
----------Lokesh Kodali-----------
$ clone_test
thread 4 is spawned
this is child   A
pid 4 clone_test: trap 14 err 5 on cpu 1 eip 0xffffffff addr 0xffffffff--kill pr
oc
zombie!
```