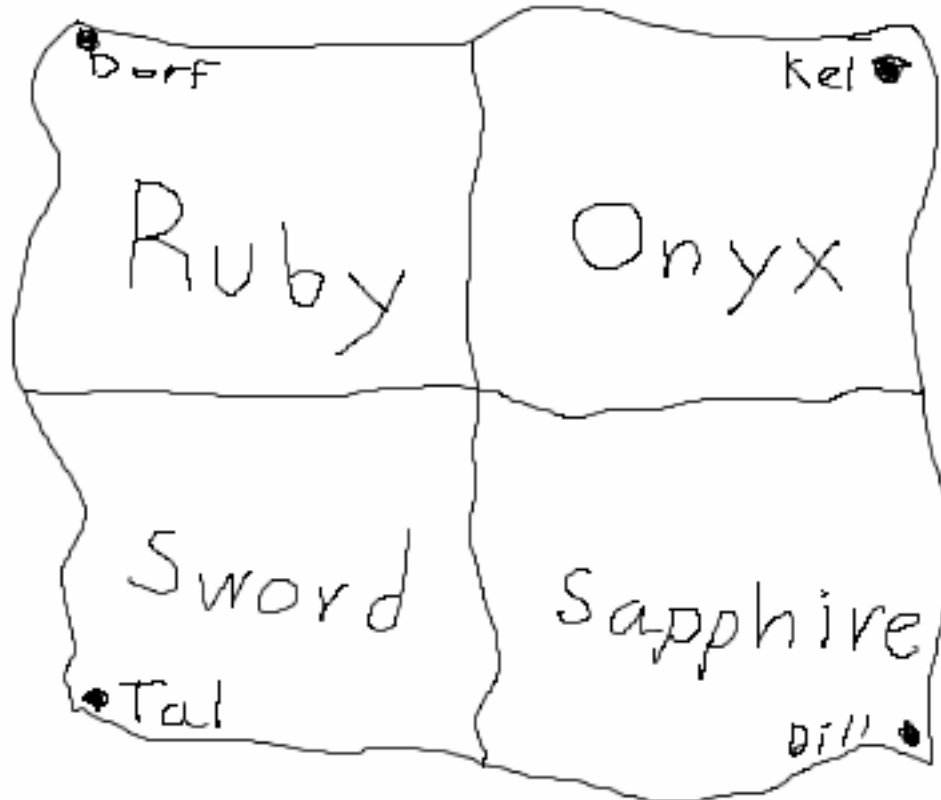


Write a LINUX program to model a hunt for treasure. An expedition consisting of four exploration teams, and a central support staff is exploring an area containing numerous caves. The exploration teams are seeking four treasures called: The White Sapphire of Charlemagne, The Black Onyx Crystal of Bohemia, The Golden Sword of Camelot, and The Great Fire Ruby of Galefrey. The three jewels are designed to fit into the handle of the sword. Once assembled, the sword is rumored to have magic healing powers, but this is probably just a legend. The expedition has come into possession of a map that narrows down the location of each treasure, but much exploration is still required. (See below.)



The treasures are located in a more-or-less rectangular region bounded by four villages, Dorf, Kel, Tal, and Dill. Each treasure will be found in a cave in the designated region. The Dorf region has 35 caves, the Kel region has 41, the Tal region has 29, and the Dill region has 33. Each cave is equally likely to contain the treasure. Some caves are larger than others. Searching a particular cave will take from one to four hours, depending on how large it is. Because of travel time, no team can search more than four caves in one day.

The program will have five threads, a central-support-staff thread, and four Exploration-Team threads, one for each region/treasure. The central-support-staff thread will start the exploration team threads, and will also start the timer. Before starting the exploration threads, it will use the *settimer* and *signal* system calls to set up a periodic timer that will issue a timer signal every 250 milliseconds. The main thread will set up a global variable that contains the current time. The current time will start with zero. Every time a signal is

received, the current time will be incremented by 1. This represents the passage of one hour of simulated time. The signal processing routine is responsible for timing the four exploration threads. Each thread will have a global variable that indicates whether it is waiting. If it is waiting, the global variable will be greater than zero. When the timer signal routine executes, it will check each of the four global variables to see if any are greater than zero. Any that are greater than zero are decremented by one. When a variable is decremented to zero, a condition variable is signaled to wake up the corresponding process. Use a single mutex variable for the entire collection of timer variables.

Base camp is at the center of the map. Each day a team will start from base camp at 9 o'clock and work for eight hours (possibly somewhat more) and return to base camp for the night. When a treasure is found, the team finding it will return immediately to base camp, inform the central-support-staff, and leave the expedition. (That is, the thread representing the team will exit.) It takes one hour for a team to reach its first cave for the day, one hour to travel between caves, and one hour for it to return to base camp. Exploration of a cave will take from one to four hours. To determine the exploration time, generate a random number from 1 to 4 using the formula  $(\text{rand}() \% 4) + 1$ . Each team-thread will keep track of the number of unexplored caves in its region. Assume that this count is kept in a variable called UC. This count must be decremented ONLY after exploring a cave. To determine if a cave contains the treasure, use a sequence of statements similar to the following:

```
if ((rand() % UC) == 0)
{
    return home
    signal central-support-staff
    exit thread
}
else
{
    UC -- ;
    Keep Searching.
}
```

If a cave does not contain the treasure, the team must go to a new cave or return to base camp. If it has been searching caves for seven hours or more, it will return to base camp, otherwise it will move to a new cave and search it. The latest that a cave-search can begin is four o'clock. No team can arrive at base camp before five o'clock. Once back at base camp, all teams that have not yet found their treasures will wait until 9 o'clock the next morning before continuing the search.

Once all treasures have been found, the central-support-staff thread will assemble the sword and exit.

Use global variables and condition variables to signal the central-support-staff and the team threads. Use a single mutex for all condition variables.

Set up the timer as follows:

```
#include <sys/time.h>
#include <signal.h>

void SignalRoutine(int)
{ // increment global variables, decrement timer variables
}

// in main
signal(SIGALRM,SignalRoutine);
struct itimerval = {{0,250000},{0,250000}};
setitimer(ITIMER_REAL,&timeval,NULL);
```

Create a thread like this:

```
pthread_t  MyThread;
...
pthread_create(MyThread,NULL,MyRoutine,Args);
...

void *MyRoutine(void *Args)
{
}
```

Exit a thread like this:

```
pthread_exit(NULL);
```

Declare mutex variables like this:

```
pthread_mutex_t  mymutex = PTHREAD_MUTEX_INITIALIZER;
```

The mutex versions of wait and signal are:

```
pthread_mutex_lock(mymutex);
pthread_mutex_unlock(mymutex);
```

Declare condition variables like this:

```
pthread_cond_t  mycondvar = PTHREAD_COND_INITIALIZER;
```

Wait on a condition like this:

```
pthread_mutex_lock(mymutex);
... (check global variables, set global variables)
pthread_cond_wait(mycondvar,mymutex);
...
pthread_mutex_unlock(mymutex);
```

Signal a condition like this:

```
pthread_mutex_lock(mymutex);  
... (check global variables)  
pthread_cond_signal(mycondvar);  
...  
pthread_mutex_unlock(mymutex);
```

Best pthreads website:

<http://www.llnl.gov/computing/tutorials/workshops/workshop/pthreads/MAIN.html>

(Lawrence Livermore Laboratory)

Print a message each time a thread does something. Here are the messages you should use:

Traveling to Cave  
Exploring Cave  
Traveling to Base Camp  
Found Onyx  
Found Sword  
Found Ruby  
Found Sapphire  
Support-Staff Got Ruby  
Support-Staff Got Sword  
Support-Staff Got Onyx  
Support-Staff Got Sapphire  
Assembling Sword  
Leaving Expedition

For the Team threads, you should identify which team is sending the message.

**Turn in:** printout of the code, printout of all messages produced in a run, E-Mail an electronic version of the code.

HINT: Don't wait for things that have already happened.

Thought question: (don't turn in) Is there a better way to manage time?