```
typedef struct p {
                                                            CS241 Lecture 20 Dining Philosophers
  pthread_mutex_t *fork_lft, *fork_rgt;
  const char *name;
  pthread t thread;
  int fail;
} Philosopher;
int running = 1;
int main()
  const char *nameList[] = { "Kant", "Guatma", "Russel", "Aristotle", "Bart" };
  pthread_mutex_t forks[5];
  Philosopher philosophers[5];
  Philosopher *phil;
  int i:
  int failed:
  for (i=0;i<5;i++) {
    failed = pthread_mutex_init(&forks[i], NULL);
    if (failed) {
      printf("Failed to initialize mutexes.");
      exit(1);
    }
  }
  for (i=0;i<5; i++) {
    phil = &philosophers[i];
    phil->name = nameList[i];
    phil->fork lft = &forks[i];
    phil->fork_rgt = &forks[(i+1)\%5];
    phil->fail = pthread_create( &phil->thread, NULL, PhilPhunction, phil);
  }
  sleep(40);
  running = 0;
  printf("cleanup time\n");
  for(i=0; i<5; i++) {
    phil = &philosophers[i];
    if (!phil->fail && pthread_join( phil->thread, NULL) ) {
```

printf("error joining thread for %s", phil->name);

exit(1);

}

return 0;

```
//http://rosettacode.org/wiki/Dining_philosophers#C
void *PhilPhunction(void *p) {
  Philosopher *phil = (Philosopher*)p;
  int failed;
  int tries_left;
  pthread_mutex_t *fork_lft, *fork_rgt, *fork_tmp;
  while (running) {
    printf("%s is sleeping --er thinking\n", phil->name);
    sleep(1+rand()\%8);
    fork_lft = phil->fork_lft;
    fork_rgt = phil->fork_rgt;
    printf("%s is hungry\n", phil->name);
    tries_left = 2; /* try twice before being forceful */
    do {
      pthread_mutex_lock( fork_lft);
      failed = (tries_left>0)? pthread_mutex_trylock( fork_rgt )
                         : pthread_mutex_lock(fork_rgt);
      if (failed) {
        pthread_mutex_unlock( fork_lft);
        fork_tmp = fork_lft;
        fork_lft = fork_rgt;
        fork_rgt = fork_tmp;
        tries left -= 1;
    } while(failed && running);
    if (!failed) {
      printf("%s is eating\n", phil->name);
      sleep(1+ rand() % 8);
      pthread_mutex_unlock( fork_rgt);
      pthread_mutex_unlock( fork_lft);
    }
  return NULL;
```