Introspection Lab Session #2

2020/11/23

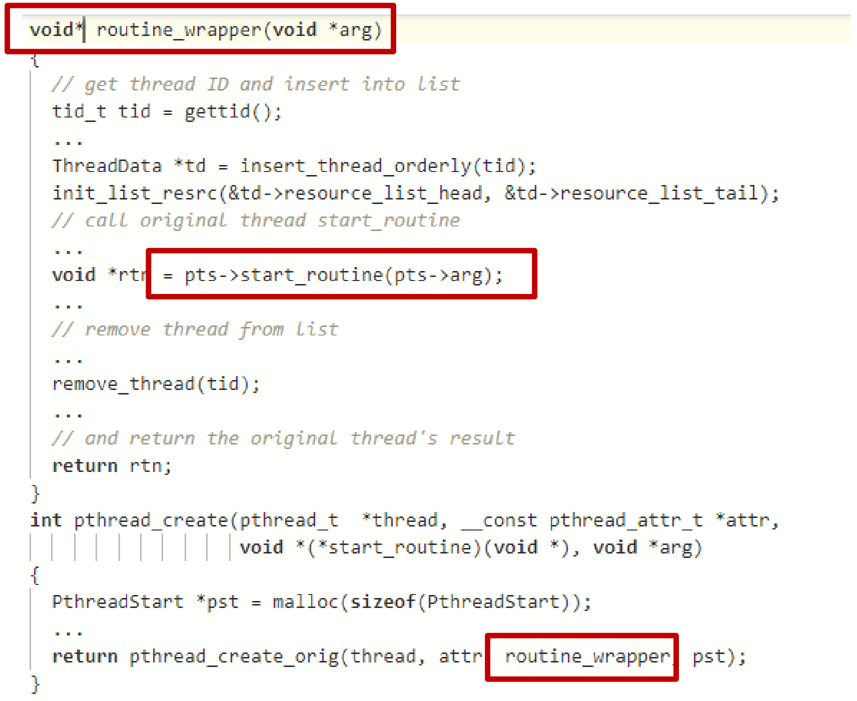
Computer Systems and Platforms Lab
Department of Computer Science and Engineering
Seoul National University

Outline

- Runtime interposition example
- Implementation of functions
- Organizing a Linked list

Implementation of functions

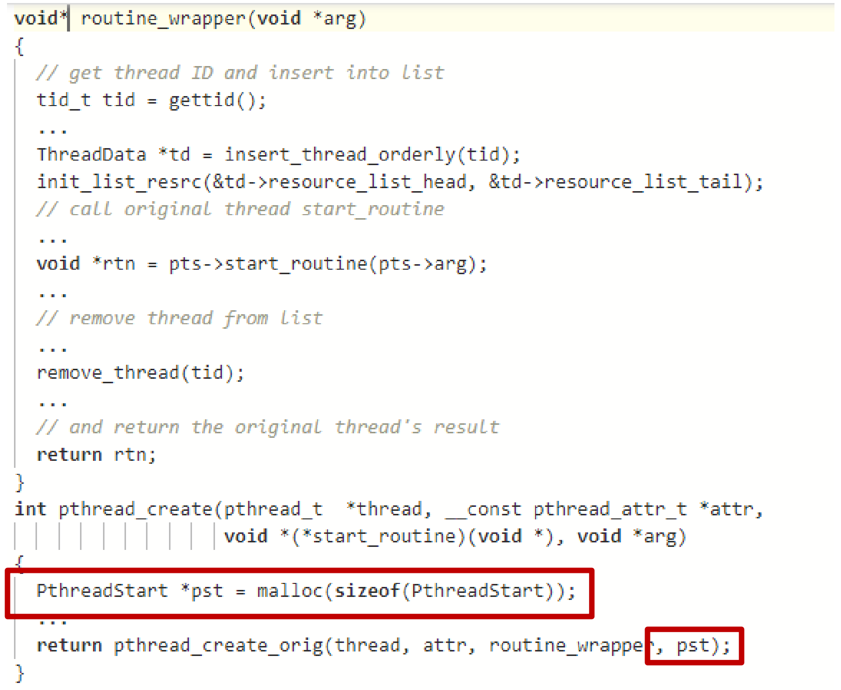
- **wrapping start_routine**



```
void* routine_wrapper(void *arg)
{
  // get thread ID and insert into list
  tid_t tid = gettid();
  ...
  ThreadData *td = insert_thread_orderly(tid);
  init_list_resrc(&td->resource_list_head, &td->resource_list_tail);
  // call original thread start_routine
  ...
  void *rtn = pts->start_routine(pts->arg);
  ...
  // remove thread from list
  ...
  remove_thread(tid);
  ...
  // and return the original thread's result
  return rtn;
}
int pthread_create(pthread_t  *thread, __const pthread_attr_t *attr,
                   void *(*start_routine)(void *), void *arg)
{
  PthreadStart *pst = malloc(sizeof(PthreadStart));
  ...
  return pthread_create_orig(thread, attr, routine_wrapper, pst);
}
```

Implementation of functions

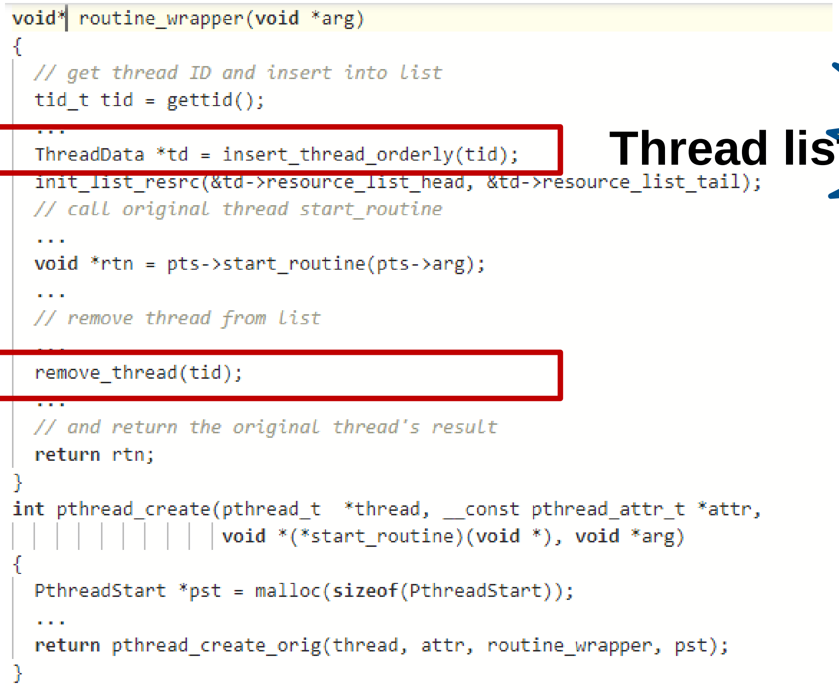
- **wrapping start_routine**



```
void* routine_wrapper(void *arg)
{
  // get thread ID and insert into list
  tid_t tid = gettid();
  ...
  ThreadData *td = insert_thread_orderly(tid);
  init_list_resrc(&td->resource_list_head, &td->resource_list_tail);
  // call original thread start_routine
  ...
  void *rtn = pts->start_routine(pts->arg);
  ...
  // remove thread from list
  ...
  remove_thread(tid);
  ...
  // and return the original thread's result
  return rtn;
}
int pthread_create(pthread_t  *thread, __const pthread_attr_t *attr,
                   void *(*start_routine)(void *), void *arg)
{
  PthreadStart *pst = malloc(sizeof(PthreadStart));
  ...
  return pthread_create_orig(thread, attr, routine_wrapper, pst);
}
```

Implementation of functions

- **wrapping start_routine**

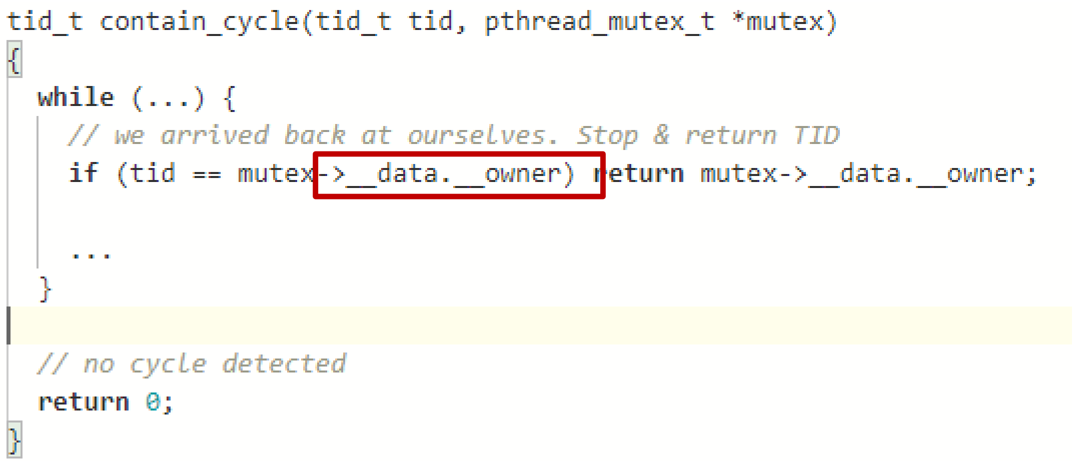


```
void* routine_wrapper(void *arg)
{
    // get thread ID and insert into list
    tid_t tid = gettid();
    ...
    ThreadData *td = insert_thread_orderly(tid);
    init_list_resrc(&td->resource_list_head, &td->resource_list_tail);
    // call original thread start_routine
    ...
    void *rtn = pts->start_routine(pts->arg);
    ...
    // remove thread from list
    ...
    remove_thread(tid);
    ...
    // and return the original thread's result
    return rtn;
}
int pthread_create(pthread_t  *thread, __const pthread_attr_t *attr,
                                  void *(*start_routine)(void *), void *arg)
{
    PthreadStart *pst = malloc(sizeof(PthreadStart));
    ...
    return pthread_create_orig(thread, attr, routine_wrapper, pst);
}
```

Thread list is shared

Implementation of functions

- **Investigating Circular Wait Condition**



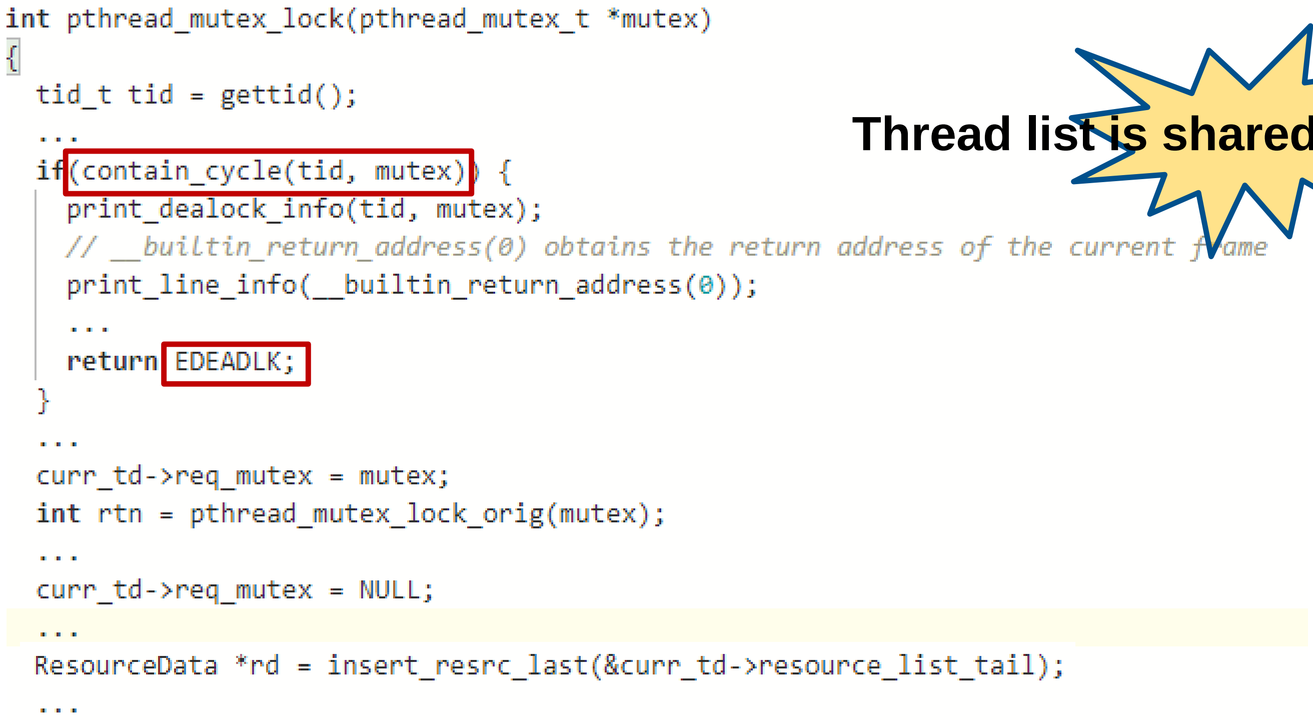
```
tid_t contain_cycle(tid_t tid, pthread_mutex_t *mutex)
{
  while (...) {
    // we arrived back at ourselves. Stop & return TID
    if (tid == mutex->__data.__owner) return mutex->__data.__owner;

    ...
  }

  // no cycle detected
  return 0;
}
```

Implementation of functions

- **Deadlock Detect**

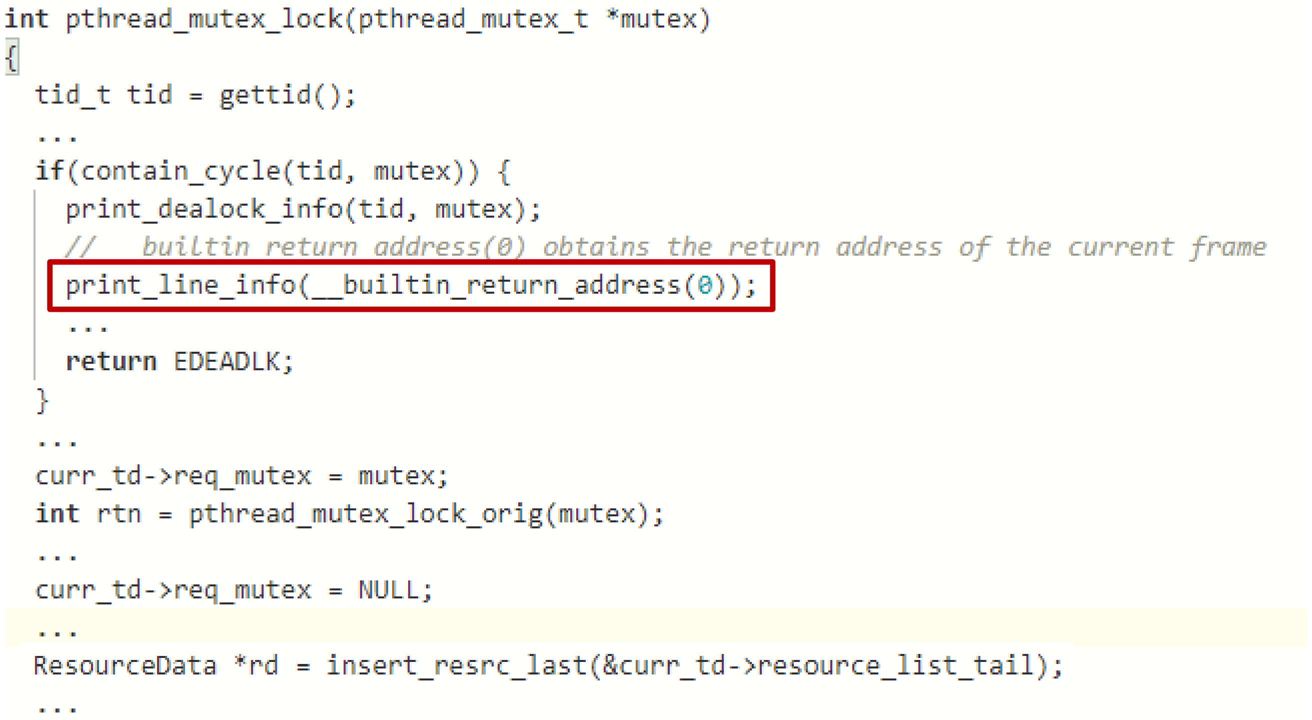


```
int pthread_mutex_lock(pthread_mutex_t *mutex)
{
  tid_t tid = gettid();
  ...
  if(contain_cycle(tid, mutex)) {
    print_dealock_info(tid, mutex);
    // __builtin_return_address(0) obtains the return address of the current frame
    print_line_info(__builtin_return_address(0));
    ...
    return EDEADLK;
  }
  ...
  curr_td->req_mutex = mutex;
  int rtn = pthread_mutex_lock_orig(mutex);
  ...
  curr_td->req_mutex = NULL;
  ...
  ResourceData *rd = insert_resrc_last(&curr_td->resource_list_tail);
  ...
```

Thread list is shared

Implementation of functions

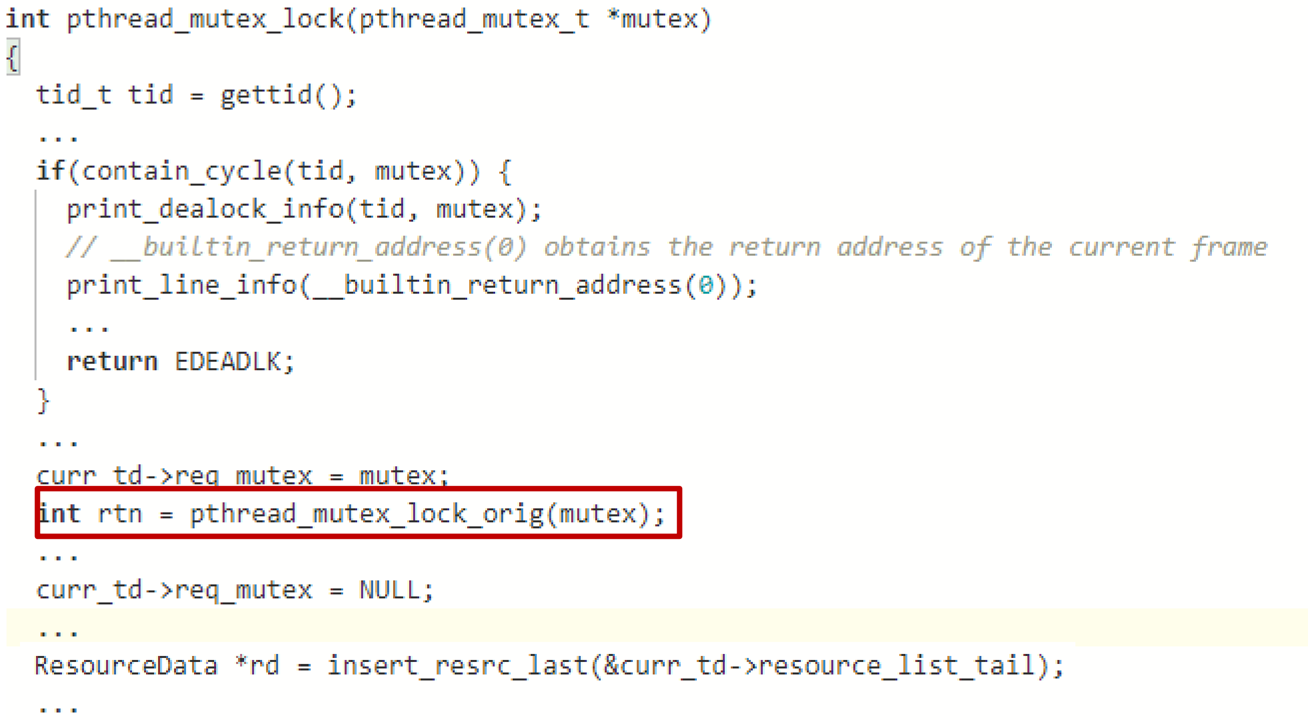
- **Deadlock Detect**



```
int pthread_mutex_lock(pthread_mutex_t *mutex)
{
  tid_t tid = gettid();
  ...
  if(contain_cycle(tid, mutex)) {
    print_dealock_info(tid, mutex);
    //   builtin return address(0) obtains the return address of the current frame
    print_line_info(__builtin_return_address(0));
    ...
    return EDEADLK;
  }
  ...
  curr_td->req_mutex = mutex;
  int rtn = pthread_mutex_lock_orig(mutex);
  ...
  curr_td->req_mutex = NULL;
  ...
  ResourceData *rd = insert_resrc_last(&curr_td->resource_list_tail);
  ...
```

Implementation of functions

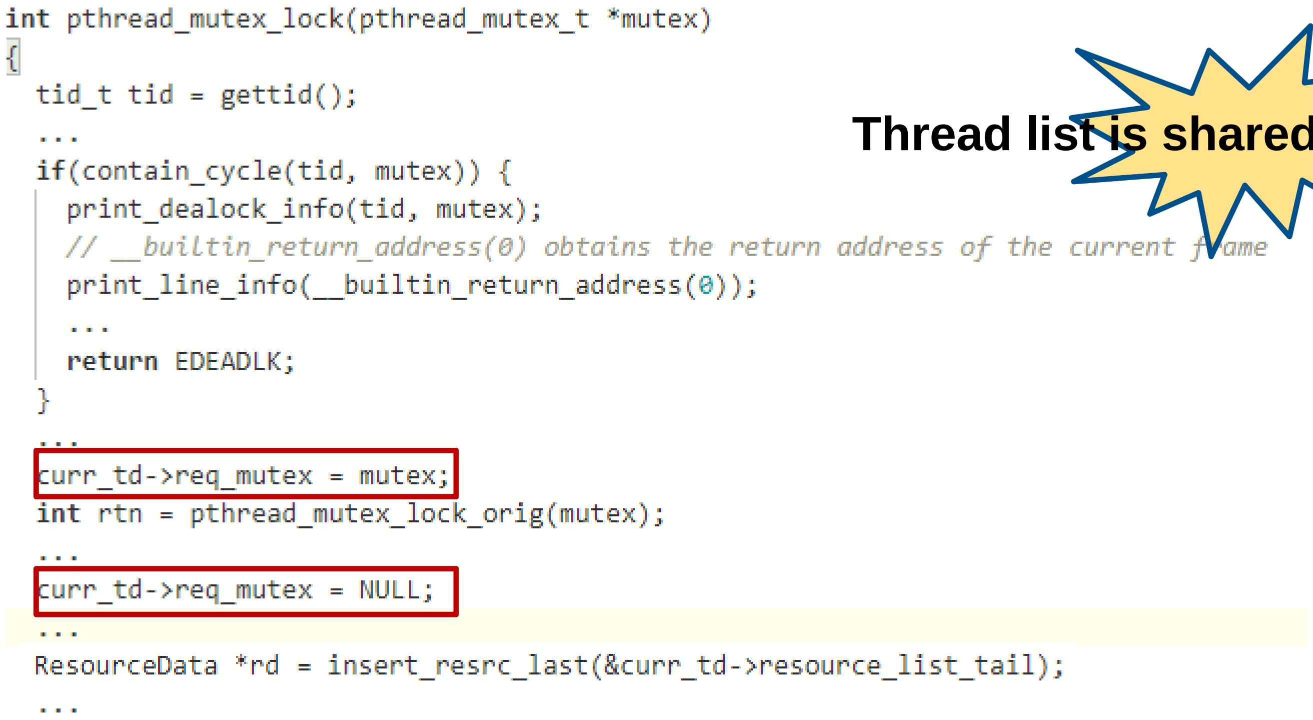
- **Deadlock Detect**



```
int pthread_mutex_lock(pthread_mutex_t *mutex)
{
  tid_t tid = gettid();
  ...
  if(contain_cycle(tid, mutex)) {
    print_dealock_info(tid, mutex);
    // __builtin_return_address(0) obtains the return address of the current frame
    print_line_info(__builtin_return_address(0));
    ...
    return EDEADLK;
  }
  ...
  curr_td->req_mutex = mutex;
  int rtn = pthread_mutex_lock_orig(mutex);
  ...
  curr_td->req_mutex = NULL;
  ...
  ResourceData *rd = insert_resrc_last(&curr_td->resource_list_tail);
  ...
```

Implementation of functions

- **Deadlock Detect**

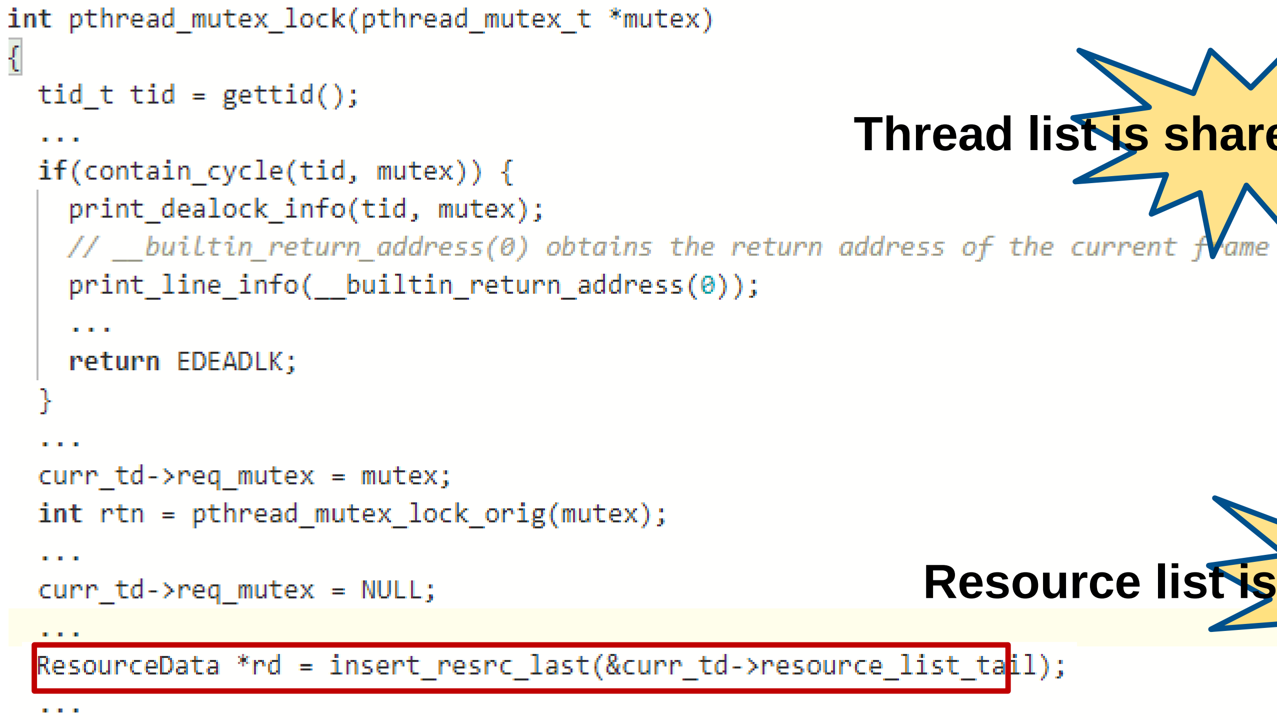


```
int pthread_mutex_lock(pthread_mutex_t *mutex)
{
  tid_t tid = gettid();
  ...
  if(contain_cycle(tid, mutex)) {
    print_dealock_info(tid, mutex);
    // __builtin_return_address(0) obtains the return address of the current frame
    print_line_info(__builtin_return_address(0));
    ...
    return EDEADLK;
  }
  ...
  curr_td->req_mutex = mutex;
  int rtn = pthread_mutex_lock_orig(mutex);
  ...
  curr_td->req_mutex = NULL;
  ...
  ResourceData *rd = insert_resrc_last(&curr_td->resource_list_tail);
  ...
```

Thread list is shared

Implementation of functions

- **Deadlock Detect**



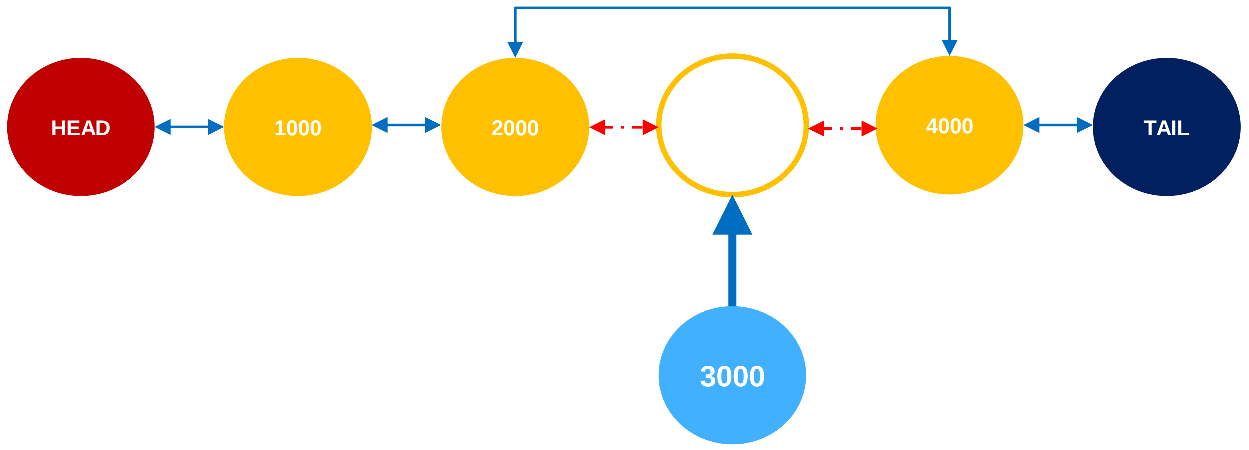
```
int pthread_mutex_lock(pthread_mutex_t *mutex)
{
  tid_t tid = gettid();
  ...
  if(contain_cycle(tid, mutex)) {
    print_dealock_info(tid, mutex);
    // __builtin_return_address(0) obtains the return address of the current frame
    print_line_info(__builtin_return_address(0));
    ...
    return EDEADLK;
  }
  ...
  curr_td->req_mutex = mutex;
  int rtn = pthread_mutex_lock_orig(mutex);
  ...
  curr_td->req_mutex = NULL;
  ...
  ResourceData *rd = insert_resrc_last(&curr_td->resource_list_tail);
  ...
```

Thread list is shared

Resource list is shared ?

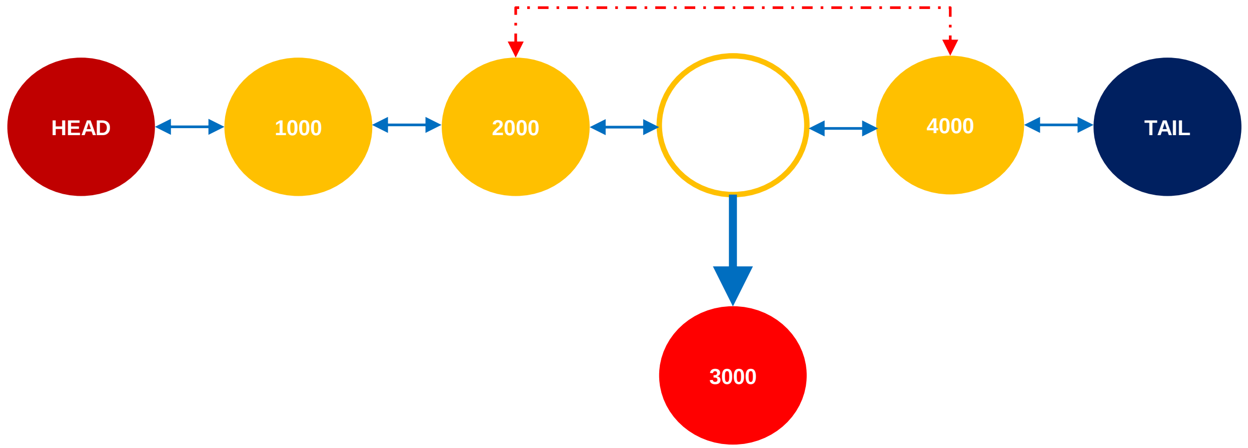
Organizing a Linked list

- ThreadData* insert_thread_orderly(tid_t tid)



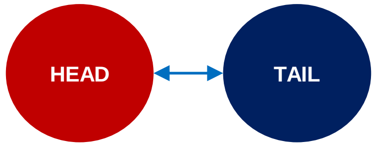
Organizing a Linked list

- void remove_thread(tid_t tid)



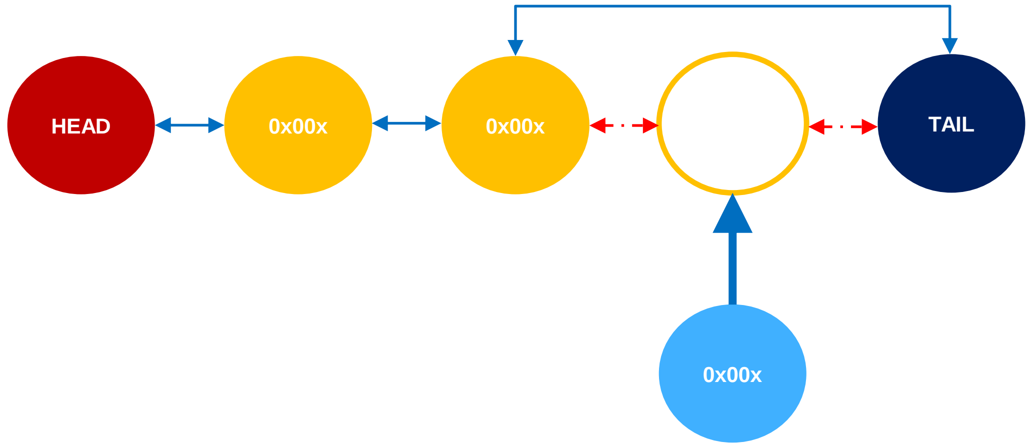
Organizing a Linked list

- void init_list_resrc(Node *head, Node *tail)



Organizing a Linked list

- ResourceData* insert_resrc_last(Node *tail)

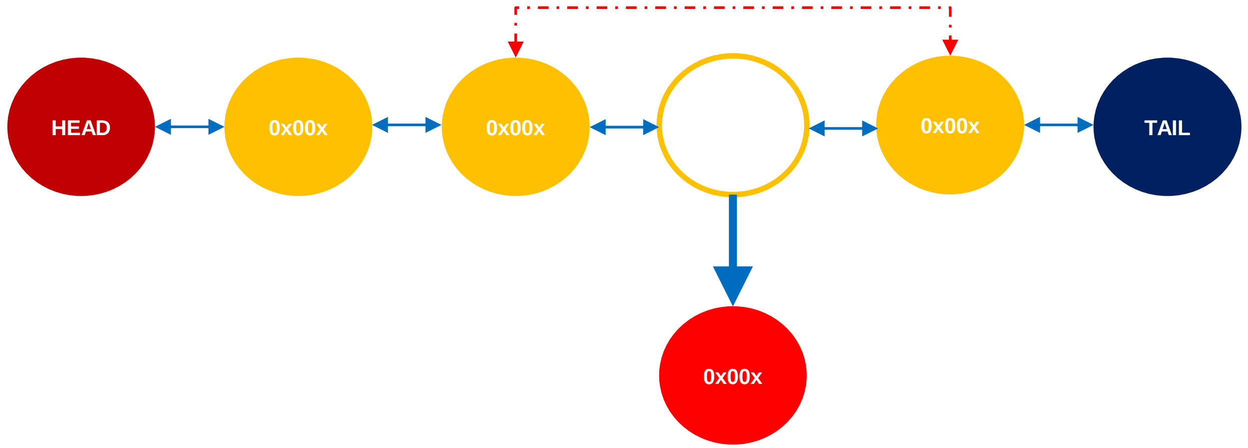


Computer Systems and Platforms Laboratory

CSE 컴퓨터공학부 Department of Computer Science & Engineering

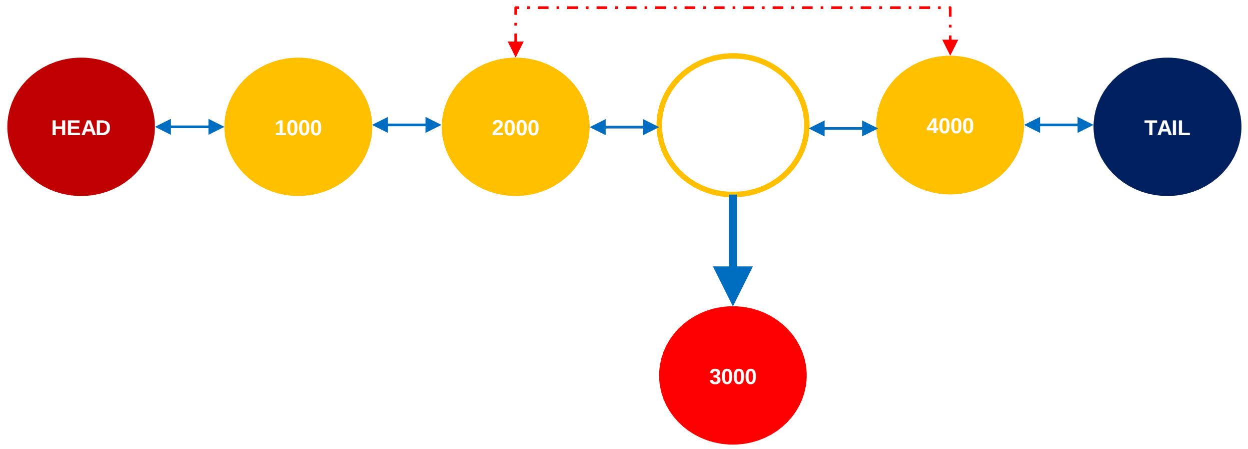
Organizing a Linked list

- void remove_resrc(Node *head, pthread_mutex_t *mutex)



Organizing a Linked list

- void remove_thread(tid_t tid)


HEAD
1000
2000
4000
TAIL
3000