1. 스케쥴러
   1. 자료구조
      1. **typedef struct \_queue {** TCB\* head;  
          TCB\* tail;  
          int size;  
          int type;  
         } queue
   2. 큐에 해당하는 API
      1. **queue\* Queue(int type)**매개변수는 큐의 종류(레디,웨잇)  
          queue.h에 #define READY\_QUEUE 1  
          queue.h에 #define WAIT\_QUEUE 2  
         반환값은 만들어진 Queue\*. 성공하면 Queue\*, 실패하면 NULL  
         레디큐 혹은 웨잇큐 생성
      2. **int Push(Queue\*, TCB\*)**매개변수는 큐에 tail에 넣을 TCB\*  
         반환값은 성공유무. 성공하면 0, 실패하면 -1  
         큐에 tail에 TCB\* 삽입.
      3. **int Pop(Queue\*)**반환값은 성공유무, 성공하면 0, 실패하면 -1  
         큐의 head 제거
      4. **TCB\* Front(Queue\*)**반환값은 큐의 head의 TCB\*. 성공하면 TCB\*반환, 실패하거나 없으면 NULL 반환  
         head 반환.
      5. **TCB\* Back(Queue\*)**반환값은 큐의 tail의 TCB\*. 성공하면 TCB\*반환, 실패하거나 없으면 NULL 반환  
         tail 반환.
      6. **Int Size(Queue\*)**반환값은 Queue의 사이즈. 성공하면 0이상, 실패하면 -1.  
         Queue의 elements의 개수를 반환.
      7. **TCB\* Search(Queue\*, thread\_t tid)**반환값은 tid에 해당하는 TCB\*. 성공하면 TCB\*, 실패하거나 없으면 NULL 반환.  
         tid에 해당하는 TCB\*를 반환한다.
      8. **int Remove(Queue\*, TCB\*)**반환값은 성공여부. 성공하면 0, 실패하면 -1  
         Queue에서 해당 TCB를 제거한다.  
         ex1) TCB를 알 경우  
          Remove(Queue, TCB\*)  
         ex2) TCB는 모르지만 tid는 알 경우  
          Remove(Queue, Search(tid))
      9. **void GetStatus(Queue\*)**Queue의 정보를 보여줌.(개수, 큐의 상태를 시각적으로 보여주는거임)  
         ex)  
         size : 4  
         front back  
         |----------|----------|----------|----------|  
         | tcb\*1 | tcb\*2 | tcb\*3 | tcb\*4 |  
          해당 tcb의 내용들주르륵
2. tc1
   1. round-robin scheduling of thread
      1. **void RunScheduler()**ppt에 알고리즘 참조(14p)
         1. rdq가 비어있다면
            1. sleep(TIME\_SLICE)
         2. rdq가 비어있지 않다면
            1. 현재 실행중인 스레드 정보  
               curtid = g\_currun
            2. 레디큐의 front 스레드 정보  
               pNewTid = rdq.front()
            3. 현재 실행중인 스레드는 레디큐 tail로, 레디큐의 front스레드는 g\_currun으로 저장
            4. \_\_ContextSwitch(curtid, pNewTid) 호출
      2. **void \_ContextSwitch(Thread aCurTid, Thread\* apNewTid)**
         1. aCurTid에 해당하는 스레드는 sleep 시킴  
            pthread\_kill(aCurTid, SIGUSR1)  
            -> \_\_thread\_wait\_handler가 등록되어 있으므로 호출 됨
         2. apNewTid에 해당하는 스레드는 깨운다  
            -> \_\_thread\_wakeup(apNewTid)
   2. Testing API scopes
      1. **thread\* \_\_getThread(pthread\_t)**Thread\* pTh = \_\_getThread(pthread\_self());  
         레디큐에서 tid에 해당되는 TCB\*를 받아오는 것.
      2. **int thread\_create(thread\_t \*thread, thread\_attr\_t \*attr, void \*(\*start\_routine) (void \*), void \*arg);**ppt에 알고리즘 참조(23,24p)
         1. pthread\_create(WrapperFunc)  
            만들어진 스레드를 WrapperFunc.funcptr에 의해 두번 대기.  
            첫번째 대기는 TCB초기화,rdq정리를 대기 : sigwait(SIGUSR1)  
            두번째 대기는 스케쥴러에 의해 실행되는 것을 대기 : \_\_thread\_wait\_handler
         2. TCB초기화
         3. rdq 뒤에 밀어넣고
         4. pthread\_kill(만들어진 스레드, SIGUSR1)  
            WrapperFunc.funcptr의 첫번째 대기를 풀어줌
3. tc2
   1. Testing API scopes
      1. **int thread\_join(thread\_t thread, void \*\*retval);**반환값은 성공여부. 성공시 0, 실패시 -1 반환  
          retval == ESRCH : 식별변호 thread가 잘못된 경우  
          retval == EINVAL : 식별번호 thread가 detached 상태일 경우
4. Tc3
   1. Testing API scopes
      1. **int thread\_suspend(thread\_t tid);**
         1. thread\_t tcb = Search(rdq, tid);
         2. tid가 rdq에 존재하면
            1. tcb의 상태 변화(READY -> SLEEP)  
               tcb.status = THREAD\_STATUS\_SLEEP;
            2. rdq에서 제거하고 wiq에 삽입  
               Remove(rdq, tcb)  
               Push(wiq, tcb)
         3. tid가 rdq에 없으면
            1. return -1
      2. **int thread\_resume(thread\_t tid);**thread\_suspend의 반대과정.  
         rdq <-> wiq 해주면 됨.
         1. thread\_t tcb = Search(wiq, tid);
         2. tid가 wiq에 존재하면
            1. tcb 상태변화(SLEEP -> READY)  
               tcb.status = THREAD\_STATUS\_READY
            2. wiq에서 제거하고 rdq에 삽입  
               Remove(wiq, tcb)  
               Push(wiq, tcb)
         3. tid가 wiq에 없으면
            1. return -1
      3. **int thread\_cancel(thread\_t tid);**반환값 ?? 모르겟음  
         thread를 강제로 종료시킴.
      4. **void thread\_exit(void\* retval);**thread\_exit을 호출한 스레드를 종료.  
         retval은 pthread\_join에서 사용 가능.
      5. **thread\* thread\_self()**