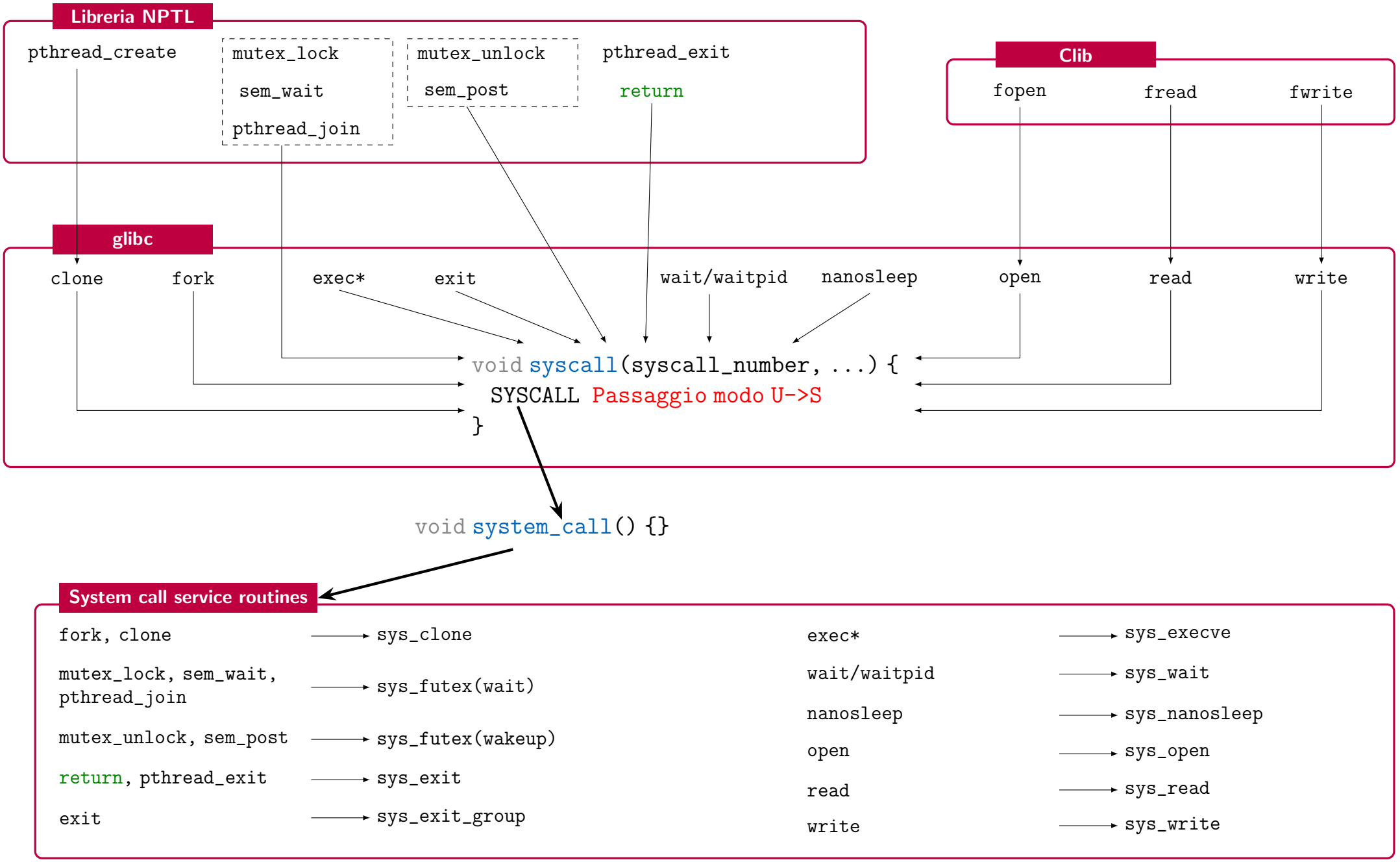
Schema chiamate sistema



Moduli del sistema operativo

System call

```
system_call(...) {
  call a servizio opportuno();
  if (modo_rientro == U &&
↪   TIF_NEED_RESCHED == 1) {
    schedule();
  }
  SYSRET
}
```

Interrupt routines

```
void R_int_clock(...) {
  // con periodicita' opportuna
  CURR->sched_class->task_tick();
  // con periodicita' opportuna
  Controlla_timer();
  if (modo_rientro == U &&
↪   TIF_NEED_RESCHED == 1) {
    schedule();
  }
  IRET
}
```

```
void R_int_X(...) {
  wakeup(...);
  if (modo_rientro == U &&
↪   TIF_NEED_RESCHED == 1) {
    schedule();
  }
  IRET
}
```

Service routines

```
sys_clone(...) {
  check_preempt_cur();
}
```

```
sys_nanosleep(...) {
  curr->state = ATTESA;
  schedule_timeout();
}
```

```
sys_futex(wait, ...) {
  wait_X(...);
}
```

```
sys_futex(wakeup, ...) {
  wake_up_process(padre);
}
```

```
sys_execve(...) {
}
```

```
sys_read/write/open(...) {
  wait_X(...);
}
```

```
sys_wait(...) {
  curr->state = ATTESA;
  schedule();
}
```

```
sys_exit(...) {
  wake_up_process(padre);
  schedule();
}
```

```
sys_exit_group(...) {
  for (tutti i processi
↪   con stesso TGID) {
    sys_exit(...)
  }
}
```

Gestione dello stato

```
wake_up/wake_up_process(...) {
  for (ogni task in waitqueue) {
    task->state = PRONTO;
    enqueue_task(task); //NN
    if (flag==EXCLUSIVE) break;
  }
  check_preempt_curr();
}
```

```
wait_X(...) {
  CURR->state = ATTESA;
  schedule();
}
```

Gestione tempo

```
schedule_timeout(...) {
  add_timer(); //NN
  schedule();
  del_timer(); //NN
}
```

```
Controlla_timer(...) {
  for (ogni timeout creato) {
    if (timeout scaduto) {
      wakeup_process(...);
    }
  }
}
```

Routine dello scheduler

```
schedule(...) {
  prev = CURR;
  if (prev->stato == ATTESA ||
      prev terminato) {
    dequeue_task(prev); //NN
  }
  next = pick_next_task();
  if (next != prev) {
    // MACRO
    context_switch(prev, next)
  }
  TIF_NEED_RESCHED = 0
}
```

```
task_tick_X(...) {
  if (QdT scaduto) {
    resched();
  }
}
```

```
check_preempt_curr(...) {
  if (occorre invocare scheduler) {
    resched();
  }
}
```

```
pick_next_task(...) {
  for (ogni classe di scheduling cl,
↪   in ordine gerarchico) {
    task = cl->pick_next_task(); //NN
    if (task) return task;
  }
}
```

```
resched(...) {
  TIF_NEED_RESCHED = 1;
}
```