1. **SJF scheduler**

   readyqueue가 비어 있지 않으면 첫번째 entry에 있는 list의 lifespan을 min_lifespan으로 지정한다.

   readyqueue를 돌면서 min_lifespan을 필요하다면 갱신하고 list를 옮긴다.

   min_lifespan이 적은 process부터 실행해야한다.

   readyqueue에서 뺄 process를 next에 저장하고 현재 돌 프로세스를 반환한다.

2. **SRTF scheduler**

   뺄 process는 sjf scheduler와 같이 구현한다.

   실행하고 있는 current의 남아있는 시간과 readyqueue에 있는 프로세스의 시간과 비교하여 current의 실행시간이 더 많이 남아 있다면 readyqueue에 다시 넣어주고 pick_next한다.

3. **RR scheduler**

   tick마다 process 진행이 이루어져야 하므로 quantum size 1일때마다 readyqueue 뒤에 다시 current를 넣어준다.

4. **Priority scheduler**

   첫번째 entry에 있는 list의 prio 값을 기준으로 지정하고 readyqueue를 돌면서 prio값이 큰 것이 있으면 그 list를 옮긴다.

   current의 prio와 readyqueue에 있는 prio와 비교하여 current의 prio가 낮으면 readyqueue에 다시 넣어준다.

5. **Priority scheduler + PCP**

   pcp_acquire : resource를 잡으면 즉시 MAX_PRIO로 올려준다. r->owner의 prio와 current의 prio를 비교하여 현재 돌고 있는 프로세스 current의 prio가 작으면 waitqueue에 넣어준다.

   pcp_release : current의 prio를 다시 원래 상태로 돌려놓는다. waitqueue가 비어있지 않으면 waiter를 waitqueue의 첫번째 entry로 지정하고 빼내어 readyqueue에 넣어 process를 진행할 준비를 한다.

   스케줄러는 prio_schedule을 사용한다.

6. Priority scheduler + PIP

pip_acquire : 높은 prio task가 낮은 prio task가 잡고있는 resource가 필요하면 낮은 priority를 가지고 도는 process의 priority를 높은 priority만큼 올린다.

pip_release : current의 prio를 다시 원래 상태로 돌려놓는다. maxPrio를 지정하고 waitqueue를 돌면서 높은 prio가 있다면 갱신하고 waitqueue에 넣어둔다. waiter를 waitqueue의 첫번째 entry로 지정하고 빼내어 readyqueue에 넣어 process를 진행할 준비를 한다.

스케줄러는 prio_schedule을 사용한다.

7. Lesson learned

list_head.h를 통해 list를 어떻게 빼고 추가하고 옮기는지 등 다양한 사용방법을 익힐 수 있었다.

이론으로만 배웠던 Process Scheduling을 직접 구현함으로써 개념을 정확히 알게 되었다.