Exercise 1

1. Address Spaces, Paging, Segmentation, Allocation, Compression, Swapping
2. The mechanism of using disk space to store and load process memory from RAM that are currently not in use. This solves the problem of more RAM requirements than physically available
3. The problem of having more RAM required than is available. Allows programs to run even when they are only partial in memory.
4. Embedded Systems, as the RAM requirements there are often small. E.g. for a washing machine or sensor nodes.
5. When accessing another processes memory space.
6. not when using paging and only the parts that are used (i.e. no unused memory).

Exercise 2

1. The $\mathcal{O}(1)$ scheduler organizes its runqueue using two arrays: "'active"' and "'expired"'. Each one contains 140 doubly linked list heads containing tasks with different priority.
   To **select**, it picks task from the highest priority queue in the active array.

   If task's time slice expires, it's **removed and inserted** into the expired list (probably in lower priority queue).
   On IO block and IO event occurs s.t. task can resume, it's placed in the same queue in the active array and its time slice is reduced to indicate the previous CPU usage.
   When slice is exhausted, task is **removed and inserted** into expired array. When no more tasks in active array, active and expired array pointers are exchanged. This prevents starvation. Different priorities are assigned different time slices. Further system keeps heuristic called dynamic priority which penalizes CPU-hogging and reward interactivity.
2. The tasks that executed previously on a CPU gets executed on the same CPU with a higher probability thus possibly avoiding cache misses. Load balancing is performed to keep the payload on all processors up and fairly the same.
3. In order to avoid using different queues for different priorities. Instead the vruntime is incremented in different steps depending on the priority and the red-black tree is sorted according to this.

Exercise 3

1. see str_rev.c

Exercise 4

1. see xor_swap.c
2. Let $a,\ b$ variables with binary base. Let $t = a$ XOR b. Thus t is 0 at place i if $a_i = b_i$ and 1 otherwise.

   $$t \text{ XOR } a = b$$
   $$t \text{ XOR } b = a$$

   is to be proven. Starting with the first conjecture.

   $$t \text{ XOR } a = a \text{ XOR } b \text{ XOR } a$$

   Using that XOR is associative and commutative, we get:

   $$a \text{ XOR } a \text{ XOR } b = 0 \text{ XOR } b = b$$

   Similarly for the second one.

Exercise 5

1. $$2 \cdot \frac{4 \text{ ns}}{32 \text{ bit}} = 2\frac{\text{ns}}{\text{bytes}}$$
   $$4 \text{ GiB} \cdot 2\frac{\text{ns}}{\text{bytes}} = 2^{32} \text{ bytes} \cdot 2 \cdot 10^{-9}\frac{\text{s}}{\text{bytes}}$$
   $$2^3 3 \cdot 10^{-9} \text{ s} = 8.589934592 \text{ s}$$
2. - First Fit: 12 MB $\mapsto$ 20 MB, 10 MB $\mapsto$ 10 MB, 9 MB $\mapsto$ 18 MB
   - Next Fit: 12 MB $\mapsto$ 20 MB, 10 MB $\mapsto$ 18 MB, 9 MB $\mapsto$ 11 MB
   - Best Fit: 12 MB $\mapsto$ 12 MB, 10 MB $\mapsto$ 10 MB, 9 MB $\mapsto$ 11 MB
   - Worst Fit: 12 MB $\mapsto$ 20 MB, 10 MB $\mapsto$ 18 MB, 9 MB $\mapsto$ 15 MB

Exercise 6

- see wl_hist.c