Scheduling

1. sched_setcheduler - Link 1
    - `int sched_setscheduler(pid_t pid, int policy, const struct sched_param *param);`
    - The sched_setscheduler() system call sets both the scheduling policy and parameters for the thread whose ID is specified in pid. If pid equals zero, the scheduling policy and parameters of the calling thread will be set.
2. sched_getscheduler - Link 1
    - `int sched_getscheduler(pid_t pid);`
    - The sched_getscheduler() function shall return the scheduling policy of the process specified by pid. If the value of pid is negative, the behavior of the sched_getscheduler() function is unspecified.
3. sched_setparam - Link 1
    - `int sched_setparam(pid_t pid, const struct sched_param *param);`
    - sched_setparam() sets the scheduling parameters associated with the scheduling policy for the thread whose thread ID is specified in pid.
4. sched_getparam - Link 1
    - `int sched_getparam(pid_t pid, struct sched_param *param);`
    - The sched_getparam() function shall return the scheduling parameters of a process specified by pid in the sched_param structure pointed to by param.
5. sched_get_priority_max - Link 1
    - `int sched_get_priority_max(int policy);`
    - sched_get_priority_max() returns the maximum priority value that an be used with the scheduling algorithm identified by policy.
6. sched_get_priority_min - Link 1
    - `int sched_get_priority_min(int policy);`
    - sched_get_priority_min() returns the minimum priority value that can be used with the scheduling algorithm identified by policy.
7. pthread_attr_setschedparam - Link 1
    - `int pthread_attr_setschedparam(pthread_attr_t *attr, struct sched_param *param);`
    - The pthread_attr_setschedparam() function sets the scheduling parameter attributes of the thread attributes object referred to by attr to the values specified in the buffer pointed to by param.
8. pthread_attr_getschedparam - Link 1
    - `int pthread_attr_getschedparam(pthread_attr_t *attr, struct sched_param *param);`
    - The pthread_attr_getschedparam() returns the scheduling parameter attributes of the thread attributes object attr in the buffer pointed to by param.

9. pthread_attr_init - Link 1

    - `int pthread_attr_init(pthread_attr_t *attr);`
    - The pthread_attr_init() function initializes the thread attributes object pointed to by attr with default attribute values.
10. pthread_attr_setschedpolicy - Link 1
    - `int pthread_attr_setschedpolicy(pthread_attr_t *attr, int policy);`
    - The pthread_attr_setschedpolicy() function sets the scheduling policy attribute of the thread attributes object referred to by attr to the value specified in policy.
11. pthread_attr_getschedpolicy - Link 1
    - `int pthread_attr_getschedpolicy(const pthread_attr_t *attr, int *policy);`
    - The pthread_attr_getschedpolicy() returns the scheduling policy attribute of the thread attributes object attr in the buffer pointed to by policy.

Some useful struct definitions:

```
struct sched_param {

      int sched_priority;

};

struct timespec {
      time_t tv_sec; // seconds
      long tv_nsec; // nanoseconds
}
```

Some important commands:

chrt -m - Lists the maximum and minimum valid priorities for all scheduling policies
chrt -p pid - Shows what the current scheduling policy and priority are for the process with process id = pid

chrt [policy flag] -p priority pid - Changes the scheduling policy of the process with process id = pid, with a priority value = priority, to the scheduling policy represented by policy flag(-o for SCHED_OTHER, -f for SCHED_FIFO, -r for SCHED_RR, -b for SCHED_BATCH, -i for SCHED_IDLE).

Eg. chrt -f -p 34 22885 - Changes the scheduling policy of the process with process id 22885 to SCHED_FIFO, with a priority value of 34.

Refer this for more details on the chrt command and for the valid range of priority. You might also need to use sudo if you are using chrt with SCHED_FIFO and SCHED_RR.

Now we will look into the concept of nice values. When you use the chrt -m command to find the maximum and minimum values of priority values for SCHED_OTHER, SCHED_BATCH and SCHED_IDLE, you will notice that they are 0 and 0. These are all non-realtime scheduling policies and it means all of these processes have the same 'priority' and their only difference is in their nice value. nice value ranges from -20 to 19. Bigger is the nice value of a process,
lower is its priority and the ‘nicer’ it is to other processes. Refer this for more details.

nice -n <nice_value> <command> - Replace <nice_value> with the priority level and <command> with the actual command or process you want to execute with that priority
renice -n <nice_value> -p <process_id> - Replace <nice_value> with the new priority level you want to assign and <process_id> with the ID of the running process you wish to modify.

Eg. renice -n 43 -p 59036 - Changes the nice value of process with process id 59036 to 43

Note: Use ps ax -o pid,ni to find the nice values of available processes.

Problem 0

Use the chrt command to find out the valid priorities of the different scheduling policies. Also use the command to find out the priority and the policy of the `bash` process.

Use the nice command to launch a process with any priority. Now, use the renice command to change the priority.