General LIBMSR Use

Before you can use any of the LIBMSR functions, you must call the init functions.

Initialize LIBMSR

1. Call init_msr, if it returns -1 there was an error opening the msr_safe or msr kernel modules.
2. If you are using RAPL, you also need to call the rapl_init function. If this returns -1 then RAPL is probably not supported on your architecture.

Initializing

```
    struct rapl_data * rd = NULL;    // Passing this to rapl_init by reference gives you one way to access the rapl data

    uint64_t * rapl_flags = NULL;    // You can pass this to rapl_init to enable/disable MSRs in case the auto-detect missed anything.
// These are both optional. See the RAPL section for more details.

    if(init_msr())

    {

        return -1;

    }

    if (rapl_init(&rd, &rapl_flags))  // If you don't need rapl data or custom flags settings, these can be NULL.

    {

        return -1;

    }
```

Finalize LIBMSR

Before you return from your main, you will want to call finalize_msr. This will close file descriptors and do other various cleanup tasks. This function also allows you to restore all MSRs to their state prior to your program's execution by passing a non-zero value to it (see bugs). This is a good idea if you are using RAPL on the clusters, to ensure the next user is not stuck with your power bounds.

Finalizing

```
     finalize_msr(1); // This will restore MSRs to their prior values
```

Related articles

- The Batch Interface
- RAPL
- General LIBMSR Use