NAME

slurm_get_end_time, slurm_get_rem_time, slurm_job_cpus_allocated_on_node, slurm_job_cpus_allocated_on_node_id, slurm_job_cpus_allocated_str_on_node, slurm_job_cpus_allocated_str_on_node_id, slurm_load_jobs, slurm_load_job_user, slurm_pid2jobid, slurm_print_job_info, slurm_print_job_info_msg − Slurm job information reporting functions

ISLURM_GET_REM_TIME, ISLURM_GET_REM_TIME2 − Fortran callable extensions

SYNTAX

```
#include <stdio.h>
#include <time.h>
#include <slurm/slurm.h>
#include <sys/types.h>

void slurm_free_job_info_msg (
        job_info_msg_t *job_info_msg_ptr
);

int slurm_load_job (
        job_info_msg_t **job_info_msg_pptr,
        uint32_t job_id,
        uint16_t show_flags,
);

int slurm_load_job_user (
        job_info_msg_t **job_info_msg_pptr,
        uint32_t user_id,
        uint16_t show_flags,
);

int slurm_load_jobs (
        time_t update_time,
        job_info_msg_t **job_info_msg_pptr,
        uint16_t show_flags
);

int slurm_notify_job (
        uint32_t job_id,
        char *message
);

int slurm_pid2jobid (
        pid_t job_pid,
        uint32_t *job_id_ptr
);

int slurm_get_end_time (
        uint32_t jobid,
        time_t *end_time_ptr
);

long slurm_get_rem_time (
        uint32_t job_id
);

void slurm_print_job_info (
        FILE *out_file,
        job_info_t *job_ptr,
        int one_liner
);

void slurm_print_job_info_msg (
        FILE *out_file,
        job_info_msg_t *job_info_msg_ptr,
        int one_liner
```

);

int **slurm_job_cpus_allocated_on_node_id** (
job_resources_t **job_resrcs_ptr*,
int *node_id*
);

int **slurm_job_cpus_allocated_on_node** (
job_resources_t **job_resrcs_ptr*,
const char **node_name*
);

int **slurm_job_cpus_allocated_str_on_node_id** (
char **cpus*,
size_t *cpus_len*,
job_resources_t **job_resrcs_ptr*,
int *node_id*
);

int **slurm_job_cpus_allocated_str_on_node** (
char **cpus*,
size_t *cpus_len*,
job_resources_t **job_resrcs_ptr*,
const char **node_name*
);

FORTRAN EXTENSION

```
INTEGER*4 JOBID, REM_TIME
REM_TIME = ISLURM_GET_REM_TIME(JOBID)
REM_TIME = ISLURM_GET_REM_TIME2()
```

ISLURM_GET_REM_TIME2() is equivalent to ISLURM_GET_REM_TIME() except that the JOBID is taken from the SLURM_JOB_ID environment variable, which is set by Slurm for tasks which it launches. Both functions return the number of seconds remaining before the job reaches the end of it's allocated time.

ARGUMENTS

cpus Specifies a pointer to allocated memory into which the string representing the list of allocated CPUs on the node is placed.

cpus_len
The size in bytes of the allocated memory space pointed by *cpus*.

data_type
Identifies the type of data to retrieve *jobinfo*. Note that different types of data are associated with different computer types and different configurations.

data The data value identified with *data_type* is returned in the location specified by *data*. See the slurm.h header file for identification of the data types associated with each value of *data_type*.

end_time_ptr
Specified a pointer to a storage location into which the expected termination time of a job is placed.

job_info_msg_pptr
Specifies the double pointer to the structure to be created and filled with the time of the last job update, a record count, and detailed information about each job. Detailed job information is written to fixed sized records and includes: ID number, name, user ID, state, assigned or requested node names, indexes into the node table, etc. In the case of indexes into the node table, this is an array of integers with pairs of start and end index number into the node information records and the data is terminated with a value of −1. See slurm.h for full details on the data structure's contents.

job_id Specifies a slurm job id. If zero, use the SLURM_JOB_ID environment variable to get the jobid.

job_id_ptr
Specifies a pointer to a storage location into which a Slurm job id may be placed.

job_info_msg_ptr
Specifies the pointer to the structure created by **slurm_load_job** or **slurm_load_jobs**.

jobinfo Job−specific information as constructed by Slurm's NodeSelect plugin. This data object is returned for each job by the **slurm_load_job** or **slurm_load_jobs** function.

job_pid Specifies a process id of some process on the current node.

job_ptr Specifies a pointer to a single job records from the *job_info_msg_ptr* data structure.

job_resrcs_ptr
Pointer to a job_resources_t structure previously using the function **slurm_load_job** with a *show_flags* value of **SHOW_DETAIL**.

node_id
Zero origin ID of a node allocated to a job.

node_name
Name of a node allocated to a job.

one_liner
Print one record per line if non−zero.

out_file Specifies the file to print data to.

show_flags
Job filtering flags, may be ORed. Information about jobs in partitions that are configured as hidden and partitions that the user's group is unable to utilize are not reported by default.

SHOW_ALL Report information about jobs in all partitions, even partitions to which the user lacks access (this access can be blocked by system administers).

SHOW_DETAIL
Report detailed resource allocation information (e.g. identification of the specific CPUs allocated to a job on each node).

SHOW_LOCAL
Report information only about jobs on the local cluster, even if the cluster is part of a federation.

SHOW_SIBLING
Report information about all sibling jobs on a federated cluster.

update_time
For all of the following informational calls, if update_time is equal to or greater than the last time changes where made to that information, new information is not returned. Otherwise all the configuration. job, node, or partition records are returned.

user_id ID of user we want information for.

DESCRIPTION

slurm_free_resource_allocation_response_msg Free slurm resource allocation response message.

slurm_free_job_info_msg Release the storage generated by the **slurm_load_jobs** function.

slurm_get_end_time Returns the expected termination time of a specified Slurm job. The time corresponds to the exhaustion of the job´s or partition´s time limit. NOTE: The data is cached locally and only retrieved from the Slurm controller once per minute.

slurm_get_rem_time Returns the number of seconds remaining before the expected termination time of a specified Slurm job id. The time corresponds to the exhaustion of the job´s or partition´s time limit. NOTE: The data is cached locally and only retrieved from the Slurm controller once per minute.

slurm_job_cpus_allocated_on_node and **slurm_job_cpus_allocated_on_node_id** return the number of CPUs allocated to a job on a specific node allocated to a job.

slurm_job_cpus_allocated_str_on_node and **slurm_job_cpus_allocated_str_on_node_id** return a string representing the list of CPUs allocated to a job on a specific node allocated to a job.

slurm_load_job Returns a job_info_msg_t that contains an update time, record count, and array of job_table records for some specific job ID.

slurm_load_jobs Returns a job_info_msg_t that contains an update time, record count, and array of job_table records for all jobs.

slurm_load_job_yser Returns a job_info_msg_t that contains an update time, record count, and array of job_table records for all jobs associated with a specific user ID.

slurm_load_job_user issues RPC to get slurm information about all jobs to be run as the specified user.

slurm_notify_job Sends the specified message to standard output of the specified job ID.

slurm_pid2jobid Returns a Slurm job id corresponding to the supplied local process id. This only works for processes which Slurm spawns and their descendants.

slurm_print_job_info Prints the contents of the data structure describing a single job records from the data loaded by the **slurm_load_node** function.

slurm_print_job_info_msg Prints the contents of the data structure describing all job records loaded by the **slurm_load_node** function.

RETURN VALUE

For **slurm_get_rem_time** on success a number of seconds is returned. For all other functions zero is returned on success. On error, −1 is returned, and Slurm error code is set appropriately.

ERRORS

SLURM_NO_CHANGE_IN_DATA Data has not changed since **update_time**.

SLURM_PROTOCOL_VERSION_ERROR Protocol version has changed, re−link your code.

ESLURM_INVALID_JOB_ID Request for information about a non−existent job.

SLURM_PROTOCOL_SOCKET_IMPL_TIMEOUT Timeout in communicating with Slurm controller.

INVAL Invalid function argument.

EXAMPLE

```
#include <stdio.h>
#include <stdlib.h>
#include <slurm/slurm.h>
#include <slurm/slurm_errno.h>
#include <sys/types.h>

int main (int argc, char *argv[])
{
        int i;
        job_info_msg_t   * job_buffer_ptr = NULL;
        job_info_t * job_ptr;
        uint32_t job_id;

        /* get and dump some job information */
        if ( slurm_load_jobs ((time_t) NULL,
                     &job_buffer_ptr, SHOW_ALL) ) {
                slurm_perror ("slurm_load_jobs error");
                exit (1);
        }

        /* The easy way to print... */
        slurm_print_job_info_msg (stdout, job_buffer_ptr, 0);
        /* A harder way.. */
        for (i = 0; i < job_buffer_ptr−>record_count; i++) {
                job_ptr = &job_buffer_ptr−>job_array[i];
                slurm_print_job_info(stdout, job_ptr, 1);
        }

        /* The hardest way. */
        printf ("Jobs updated at %lx, record count %d\n",
            job_buffer_ptr−>last_update,
            job_buffer_ptr−>record_count);
        for (i = 0; i < job_buffer_ptr−>record_count; i++) {
                printf ("JobId=%u UserId=%u\n",
                        job_buffer_ptr−>job_array[i].job_id,
                        job_buffer_ptr−>job_array[i].user_id);
        }

        slurm_free_job_info_msg (job_buffer_ptr);

        if (slurm_pid2jobid (getpid(), &job_id))
                slurm_perror ("slurm_load_jobs error");
        else
                printf ("Slurm job id = %u\n", job_id);

        exit (0);
}
```

NOTES

These functions are included in the libslurm library, which must be linked to your process for use (e.g. "cc −lslurm myprog.c").

The *command* field in the job record will be the name of user program to be launched by the srun or sbatch command. The field is not set when either the salloc command is used or the sbatch command is used with the −−wrap option.

Some data structures contain index values to cross−reference each other. If the *show_flags* argument is not set to SHOW_ALL when getting this data, these index values will be invalid.

The **slurm_hostlist_** functions can be used to convert Slurm node list expressions into a collection of individual node names.

COPYING

SEE ALSO

scontrol(1), **squeue**(1), **slurm_hostlist_create**(3), **slurm_hostlist_shift**(3), **slurm_hostlist_destroy**(3), **slurm_allocation_lookup**(3), **slurm_get_errno**(3), **slurm_perror**(3), **slurm_strerror**(3)