NAME

slurm_free_node_info_msg, slurm_load_node, slurm_load_node_single, slurm_print_node_info_msg, slurm_print_node_table, slurm_sprint_node_table − Slurm node information reporting functions

SYNTAX

#include <stdio.h>
#include <slurm/slurm.h>

void **slurm_free_node_info_msg** (
node_info_msg_t **node_info_msg_ptr*
);

int **slurm_load_node** (
time_t *update_time*,
node_info_msg_t ***node_info_msg_pptr*,
uint16_t *show_flags*
);

int **slurm_load_node_single** (
node_info_msg_t ***node_info_msg_pptr*,
char **node_name*,
uint16_t *show_flags*
);

void **slurm_print_node_info_msg** (
FILE **out_file*,
*node_info_msg_t *node_info_msg_ptr,*
int one_liner
);

void **slurm_print_node_table** (
FILE **out_file*,
*node_info_t *node_ptr,*
int one_liner
);

char ***slurm_sprint_node_table** (
node_info_t **node_ptr*,
int *one_liner*
);

ARGUMENTS

node_info_msg_ptr
Specifies the pointer to the structure created by **slurm_load_node**.

node_info_msg_pptr
Specifies the double pointer to the structure to be created and filled with the time of the last node update, a record count, and detailed information about each node. Detailed node information is written to fixed sized records and includes: name, state, processor count, memory size, etc. See slurm.h for full details on the data structure's contents.

node_name
Name of the node for which information is requested.

node_ptr
Specifies a pointer to a single node record from the *node_info_msg_ptr* data structure.

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 nodes in partitions that are configured as hidden and partitions that the user's group is unable to utilize are not reported by default. The **SHOW_ALL** flag will cause information about nodes in all partitions to be displayed. Only information about nodes on the local cluster will be returned unless the cluster is in a

federation and the **SHOW_ALL** flag is set.

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.

DESCRIPTION

slurm_free_node_info_msg Release the storage generated by the **slurm_load_node** function.

slurm_load_node_single issue RPC to get slurm configuration information for a specific node.

slurm_load_node Returns a *node_info_msg_t* that contains an update time, record count, and array of node_table records for all nodes. Note that nodes which are hidden for any reason will have a NULL node name. Other fields associated with the node will be filled in appropriately. Reasons for a node being hidden include: a node state of FUTURE, a node in the CLOUD that is powered down, or a node in a hidden partition.

slurm_print_node_info_msg Prints the contents of the data structure describing all node records from the data loaded by the **slurm_load_node** function.

slurm_print_node_table Prints the contents of the data structure describing a single node record loaded by the **slurm_load_node** function.

RETURN VALUE

On success, zero is returned. 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.

SLURM_PROTOCOL_SOCKET_IMPL_TIMEOUT Timeout in communicating with Slurm controller.

EXAMPLE

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

int main (int argc, char *argv[])
{
        int i, j, k;
        partition_info_msg_t *part_buffer_ptr = NULL;
        partition_info_t *part_ptr;
        node_info_msg_t *node_buffer_ptr = NULL;
        node_info_t *node_ptr;

        /* get and dump some node information */
        if ( slurm_load_node ((time_t) NULL,
                        &node_buffer_ptr, SHOW_ALL) ) {
                slurm_perror ("slurm_load_node error");
                exit (1);
        }

        /* The easy way to print... */
        slurm_print_node_info_msg (stdout, node_buffer_ptr, 0);

        /* A harder way.. */
        for (i = 0; i < node_buffer_ptr->record_count; i++) {
                node_ptr = &node_buffer_ptr->node_array[i];
                slurm_print_node_table(stdout, node_ptr, 0, 0);
        }

        /* The hardest way. */
        for (i = 0; i < node_buffer_ptr->record_count; i++) {
```

```
                printf ("NodeName=%s CPUs=%u\n",
                        node_buffer_ptr->node_array[i].name,
                        node_buffer_ptr->node_array[i].cpus);
        }

        /* get and dump some partition information */
        /* note that we use the node information loaded */
        /* above and we assume the node table entries have */
        /* not changed since */
        if ( slurm_load_partitions ((time_t) NULL,
                        &part_buffer_ptr, 0) ) {
                slurm_perror ("slurm_load_partitions error");
                exit (1);
        }
        for (i = 0; i < part_buffer_ptr->record_count; i++) {
                part_ptr = &part_buffer_ptr->partition_array[i];
                printf ("PartitionName=%s Nodes=",
                        part_ptr->name);
                for (j = 0; part_ptr->node_inx; j+=2) {
                        if (part_ptr->node_inx[j] == -1)
                                break;
                        for (k = part_ptr->node_inx[j];
                            k <= part_ptr->node_inx[j+1];
                            k++) {
                                printf ("%s ", node_buffer_ptr->
                                      node_array[k].name);
                        }
                }
                printf("\n\n");
        }
        slurm_free_node_info_msg (node_buffer_ptr);
        slurm_free_partition_info_msg (part_buffer_ptr);
        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").

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.

COPYING

SEE ALSO

scontrol(1), **squeue**(1), **slurm_allocation_lookup**(3), **slurm_get_errno**(3), **slurm_load_partitions**(3), **slurm_perror**(3), **slurm_strerror**(3)