NAME

slurm_checkpoint_able, slurm_checkpoint_complete, slurm_checkpoint_create, slurm_checkpoint_disable, slurm_checkpoint_enable, slurm_checkpoint_error, slurm_checkpoint_restart, slurm_checkpoint_vacate − Slurm checkpoint functions

SYNTAX

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

int slurm_checkpoint_able (
        uint32_t job_id,
        uint32_t step_id,
        time_t *start_time,
);

int slurm_checkpoint_complete (
        uint32_t job_id,
        uint32_t step_id,
        time_t start_time,
        uint32_t error_code,
        char *error_msg
);

int slurm_checkpoint_create (
        uint32_t job_id,
        uint32_t step_id,
        uint16_t max_wait,
        char *image_dir
);

int slurm_checkpoint_disable (
        uint32_t job_id,
        uint32_t step_id
);

int slurm_checkpoint_enable (
        uint32_t job_id,
        uint32_t step_id
);

int slurm_checkpoint_error (

        uint32_t job_id,
        uint32_t step_id,
        uint32_t *error_code,
        char ** error_msg
);

int slurm_checkpoint_restart (
        uint32_t job_id,
        uint32_t step_id,
        uint16_t stick,
        char *image_dir
);

int slurm_checkpoint_tasks (
        uint32_t job_id,
        uint32_t step_id,
        time_t begin_time,
        char *image_dir,
        uint16_t max_wait,
        char *nodelist
);

int slurm_checkpoint_vacate (
```

uint32_t *job_id*,
uint32_t *step_id*,
uint16_t *max_wait*,
char **image_dir*
);

ARGUMENTS

begin_time
When to begin the operation.

error_code
Error code for checkpoint operation. Only the highest value is preserved.

error_msg
Error message for checkpoint operation. Only the *error_msg* value for the highest *error_code* is preserved.

image_dir
Directory specification for where the checkpoint file should be read from or written to. The default value is specified by the *JobCheckpointDir* Slurm configuration parameter.

job_id Slurm job ID to perform the operation upon.

max_wait
Maximum time to allow for the operation to complete in seconds.

nodelist
Nodes to send the request.

start_time
Time at which last checkpoint operation began (if one is in progress), otherwise zero.

step_id Slurm job step ID to perform the operation upon. May be NO_VAL if the operation is to be performed on all steps of the specified job. Specify SLURM_BATCH_SCRIPT to checkpoint a batch job.

stick If non−zero then restart the job on the same nodes that it was checkpointed from.

DESCRIPTION

slurm_checkpoint_able Report if checkpoint operations can presently be issued for the specified job step. If yes, returns SLURM_SUCCESS and sets *start_time* if checkpoint operation is presently active. Returns ESLURM_DISABLED if checkpoint operation is disabled.

slurm_checkpoint_complete Note that a requested checkpoint has been completed.

slurm_checkpoint_create Request a checkpoint for the identified job step. Continue its execution upon completion of the checkpoint.

slurm_checkpoint_disable Make the identified job step non−checkpointable. This can be issued as needed to prevent checkpointing while a job step is in a critical section or for other reasons.

slurm_checkpoint_enable Make the identified job step checkpointable.

slurm_checkpoint_error Get error information about the last checkpoint operation for a given job step.

slurm_checkpoint_restart Request that a previously checkpointed job resume execution. It may continue execution on different nodes than were originally used. Execution may be delayed if resources are not immediately available.

slurm_checkpoint_vacate Request a checkpoint for the identified job step. Terminate its execution upon completion of the checkpoint.

RETURN VALUE

Zero is returned upon success. On error, −1 is returned, and the Slurm error code is set appropriately.

ERRORS

ESLURM_INVALID_JOB_ID the requested job or job step id does not exist.

ESLURM_ACCESS_DENIED the requesting user lacks authorization for the requested action (e.g. trying to delete or modify another user’s job).

ESLURM_JOB_PENDING the requested job is still pending.

ESLURM_ALREADY_DONE the requested job has already completed.

ESLURM_DISABLED the requested operation has been disabled for this job step. This will occur when a request for checkpoint is issued when they have been disabled.

ESLURM_NOT_SUPPORTED the requested operation is not supported on this system.

EXAMPLE

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

int main (int argc, char *argv[])
{
        uint32_t job_id, step_id;

        if (argc < 3) {
                printf("Usage: %s job_id step_id\n", argv[0]);
                exit(1);
        }

        job_id = atoi(argv[1]);
        step_id = atoi(argv[2]);
        if (slurm_checkpoint_disable(job_id, step_id)) {
                slurm_perror ("slurm_checkpoint_error:");
                exit (1);
        }
        exit (0);
}
```

NOTE

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

COPYING

SEE ALSO

srun(1), **squeue**(1), **free**(3), **slurm.conf**(5)