NAME

libscamperctrl — scamper control library

LIBRARY

scamper control library (libscamperctrl -lscamperctrl)

SYNOPSIS

```
#include <libscamperctrl.h>
```

DESCRIPTION

The **libscamperctrl** library provides an application programming interface (API) to execute measurements using the control mechanisms available for a set of external scamper(1) processes.

ROUTINES

scamper_ctrl_t * **scamper_ctrl_alloc**(*scamper_ctrl_cb_t cb*)
Allocate a control structure for managing a set of scamper(1) instances. The caller must pass a callback function, which must be defined as follows:

void **cb**(*scamper_inst_t *inst*, *uint8_t type*, *scamper_task_t *task*, *const void *data*, *size_t len*)

scamper_ctrl_wait() calls this function every time there is an event. It passes a pointer to the scamper instance that the event is in relation to in the first parameter. The second parameter identifies the type of the event, which could be one of the following.

- **SCAMPER_CTRL_TYPE_DATA:** a measurement task has completed. The task pointer points to the task returned by **scamper_inst_do**(). The data pointer points to the data, and the length parameter reports the number of bytes of data. The programmer does not have to call **scamper_task_free**() unless they kept a copy of the task pointer by calling **scamper_task_use**().
- **SCAMPER_CTRL_TYPE_MORE:** the scamper instance is requesting another command to execute.
- **SCAMPER_CTRL_TYPE_ERR:** the command was not accepted.
- **SCAMPER_CTRL_TYPE_EOF:** the scamper instance disconnected.
- **SCAMPER_CTRL_TYPE_FATAL:** the library encountered a fatal error.

int **scamper_ctrl_wait**(*scamper_ctrl_t *ctrl*, *struct timeval *to*)
Wait for events from the collection of scamper(1) instances. If the timeout parameter is null, then the wait function will block indefinitely until an event occurs. Otherwise, this function will return after the length of time specified in the timeout parameter has elapsed, or after it has called the callback function for an event.

void **scamper_ctrl_free**(*scamper_ctrl_t *ctrl*)
Cleanup the resources associated with managing the set of scamper(1) instances.

int **scamper_ctrl_isdone**(*scamper_ctrl_t *ctrl*)
Returns 0 if there is at least one active scamper instance. Returns non-zero if there are no active scamper instances.

scamper_inst_t * **scamper_inst_unix**(*scamper_ctrl_t *ctrl*, *const char *path*)
Attach to a local scamper instance via the unix domain socket at the supplied path.

scamper_inst_t * **scamper_inst_inet**(*scamper_ctrl_t *ctrl*, *const char *addr*, *uint16_t port*)
Attach to a local scamper instance via the supplied IP address and port.

scamper_inst_t * **scamper_inst_remote**(*scamper_ctrl_t *ctrl*, *const char *path*)
Attach to a remote scamper instance via the unix domain socket at the supplied path. The remote scamper

instance is connected to the local machine using a `sc_remoted`(3) process.

void **scamper_inst_free**(*scamper_inst_t *inst*)
Disconnect and then free the resources associated with the instance.

scamper_task_t * **scamper_inst_do**(*scamper_inst_t *inst*, *const char *cmd*)
Issue a command on the supplied instance. The returned task pointer uniquely identifes the task. If a programmer wishes to keep a copy of the task pointer in the process, they must call **scamper_task_use**() to keep a reference, and then call **scamper_task_free**() once done with it.

int **scamper_inst_halt**(*scamper_inst_t *inst*, *scamper_task_t *task*)
Issue a halt command for the task. If the command has not been issued to scamper yet, then no DATA object will be returned; otherwise, a DATA object will be returned.

int **scamper_inst_done**(*scamper_inst_t *inst*)
Send a done command, which will cause the `scamper`(1) instance to disconnect when it has returned all completed measurements.

void **scamper_task_free**(*scamper_task_t *task*)
Free task resources, if the programmer previously called **scamper_task_use**() to advice `libscamperctrl`(3) that the programmer would also be keeping a copy of a task pointer.

void **scamper_task_use**(*scamper_task_t *task*)
Advise `libscamperctrl`(3) that the programmer is keeping a copy of the task in their program.

SEE ALSO

`scamper`(1), `sc_remoted`(1), `libscamperfile`(3),

M. Luckie, *Scamper: a Scalable and Extensible Packet Prober for Active Measurement of the Internet*, Proc. ACM/SIGCOMM Internet Measurement Conference 2010.

AUTHORS

libscamperctrl was written by Matthew Luckie <mjl@luckie.org.nz>.