UCLouvain

Plateforme technologique de Calcul Intensif et Stockage de Masse

EURO BELGIUM

Checkpointing

Olivier Mattelaer

Slides from Damien Francois

What is checkpointing

?

```
$ ./count
$ ./count
1
```

```
$ ./count
1
2
```

```
$ ./count
1
2
3
```

```
$ ./count
1
2
3^C
$
```

```
$ ./count
1
2
3^C
$ ./count
$ ./count
1
2
3^C
$ ./count
1
```

Without checkpointing:

```
$ ./count
1
2
3^C
$ ./count
1
```

Without checkpointing:

```
$ ./count
1
2
3^C
$ ./count
1
```

With checkpointing:

```
$ ./count
1
2
3^C
$ ./count
4
```

Without checkpointing:

```
$ ./count
1
2
3^C
$ ./count
1
2
```

With checkpointing:

```
$ ./count
1
2
3^C
$ ./count
4
5
```

Without checkpointing:

```
$ ./count
1
2
3^C
$ ./count
1
2
3
```

With checkpointing:

```
$ ./count
1
2
3^C
$ ./count
4
5
6
```

Checkpointing:

'saving' a computation
so that it can be resumed later
(rather than started again)

Without checkpointing:

```
$ ./count
1
2
3^C
$ ./count
1
2
3
```

With checkpointing:

```
$ ./count
1
2
3^C
$ ./count
4
5
6
```

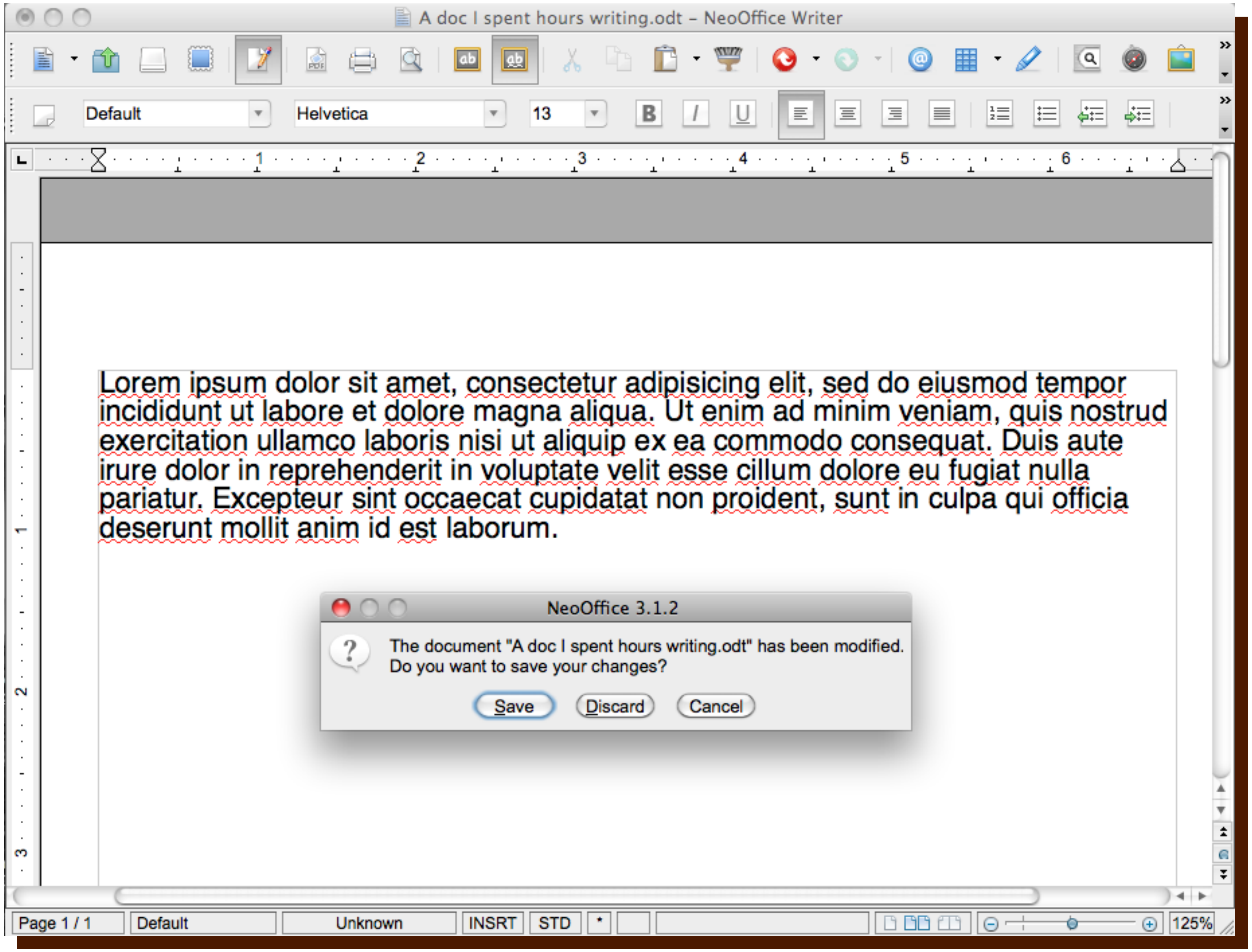
Today's agenda:

1. General concepts and scientific soft.
2. Working with Signals
3. Slurm recipes
4. DMTCP

Why do we need checkpointing

?

Imagine a text editor without 'checkpointing' ...



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

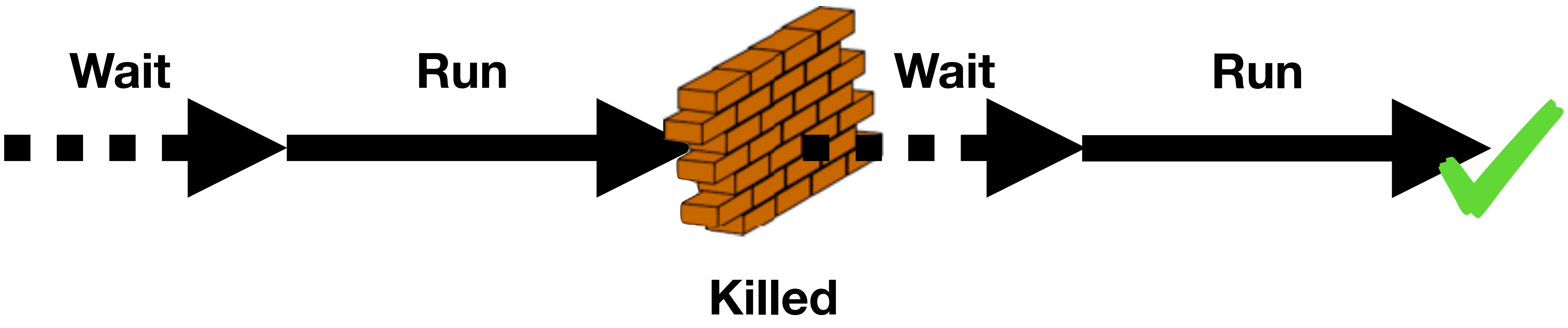
NeoOffice 3.1.2

The document "A doc I spent hours writing.odt" has been modified.
Do you want to save your changes?

Goals of checkpointing in HPC:

1. Fit in time constraints
2. Debugging, monitoring
3. Cope with hardware failures
4. Job preemption

Wall-Time



Wait

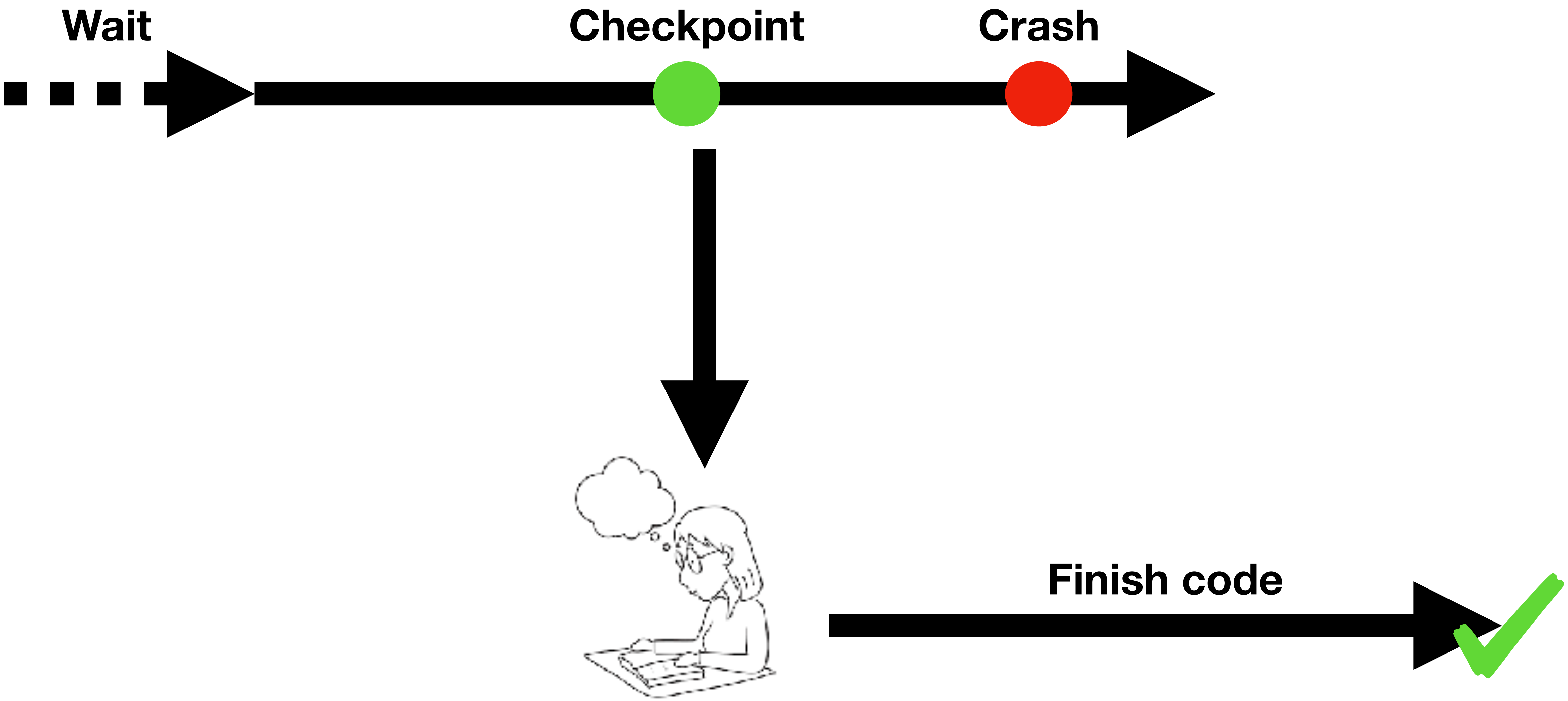
Run

Wait

Run

Killed

Debug



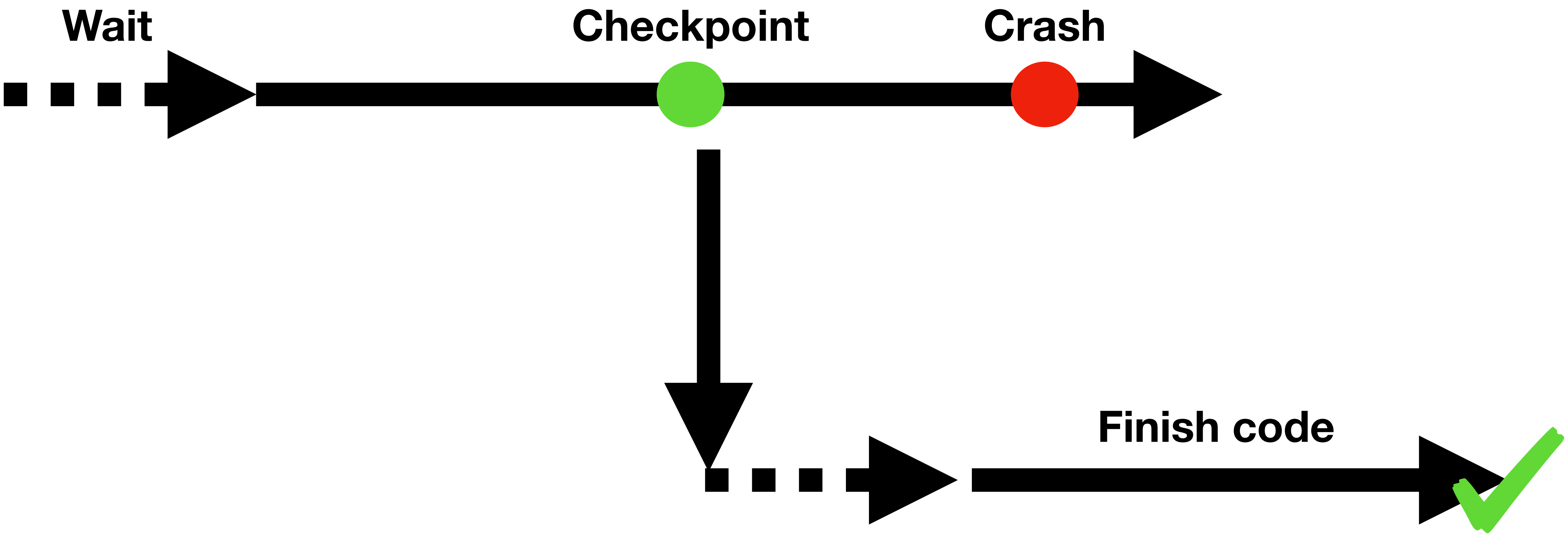
Wait

Checkpoint

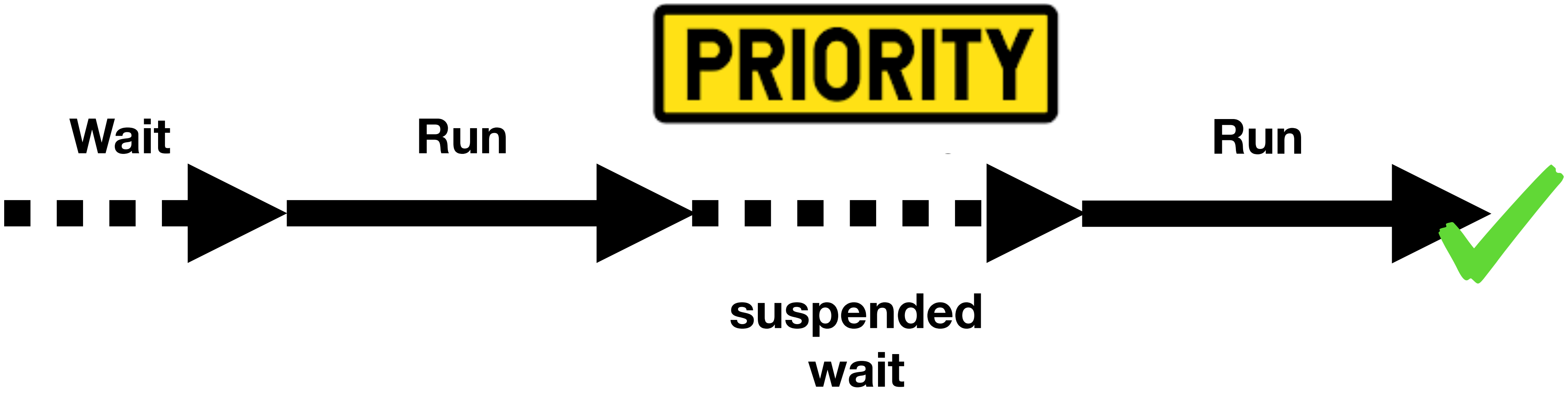
Crash

Finish code

Hardware crash



Pre-emption



PRIORITY

Wait

Run

Run

suspended wait

1 Checkpointing with scientific software
Do they support checkpointing?

Working with checkpoint-restart-able software

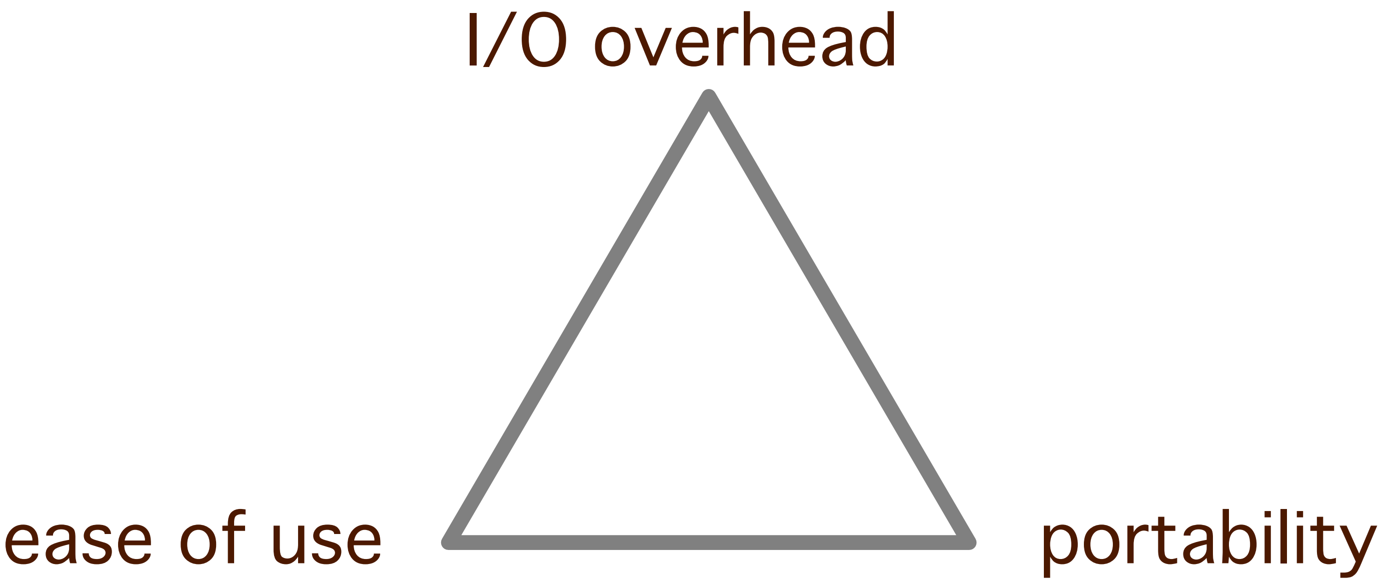
Many scientific software have built-in checkpointing capabilities

(although it might not be called that way)

Check the documentation

Need to implement it yourself?

Evaluate the options : tradeoff between



Transparency for developer	Portability to other systems	Size of state to save	Checkpointing overhead
Do I need to write a lot of additional code ?	Can I stop on one system and restart on another ?	How many GB of disk does it require ?	How many FLOPs lost to ensure checkpointing ?

Demo #1

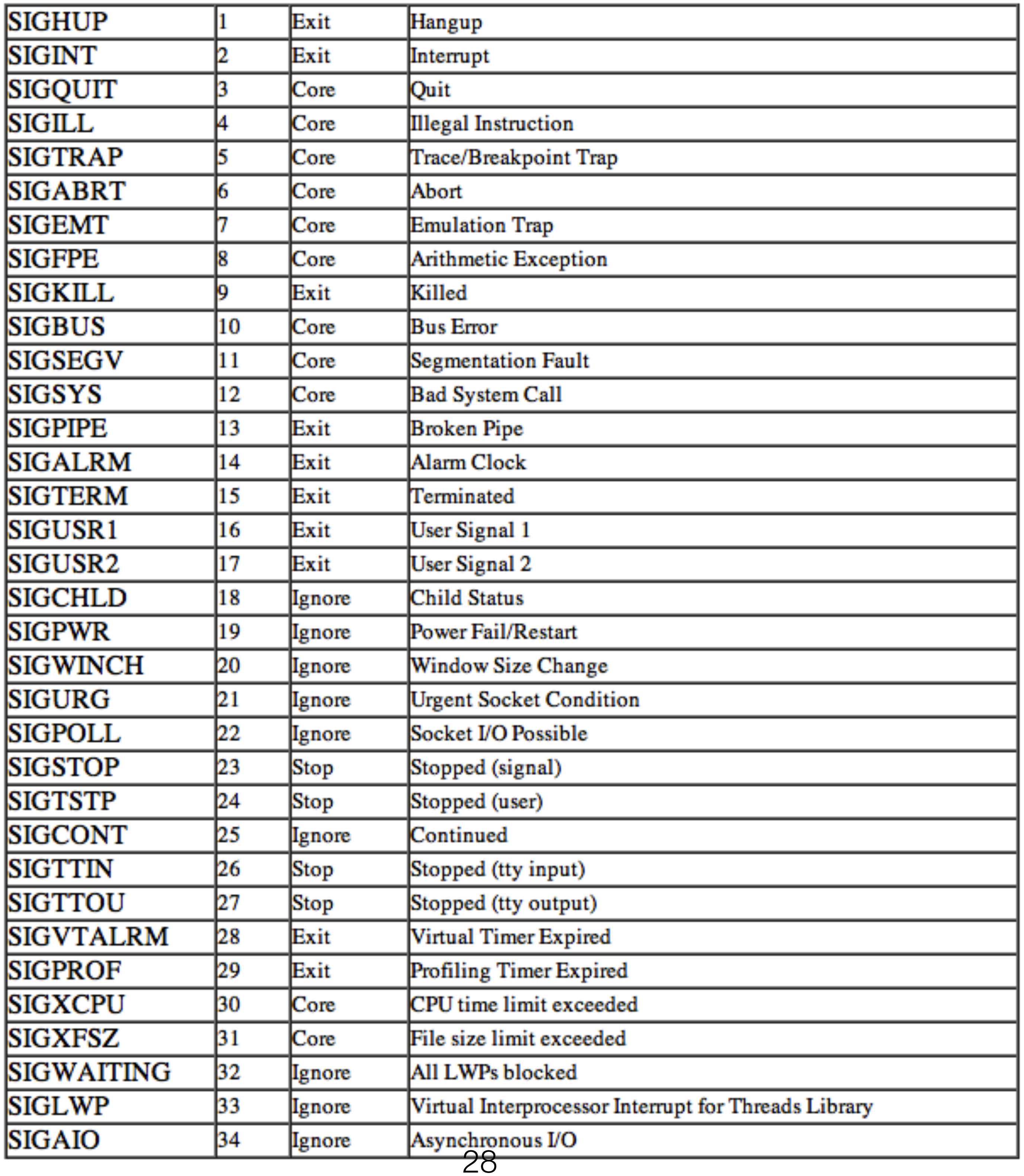
count.py

Save state at each iteration

2

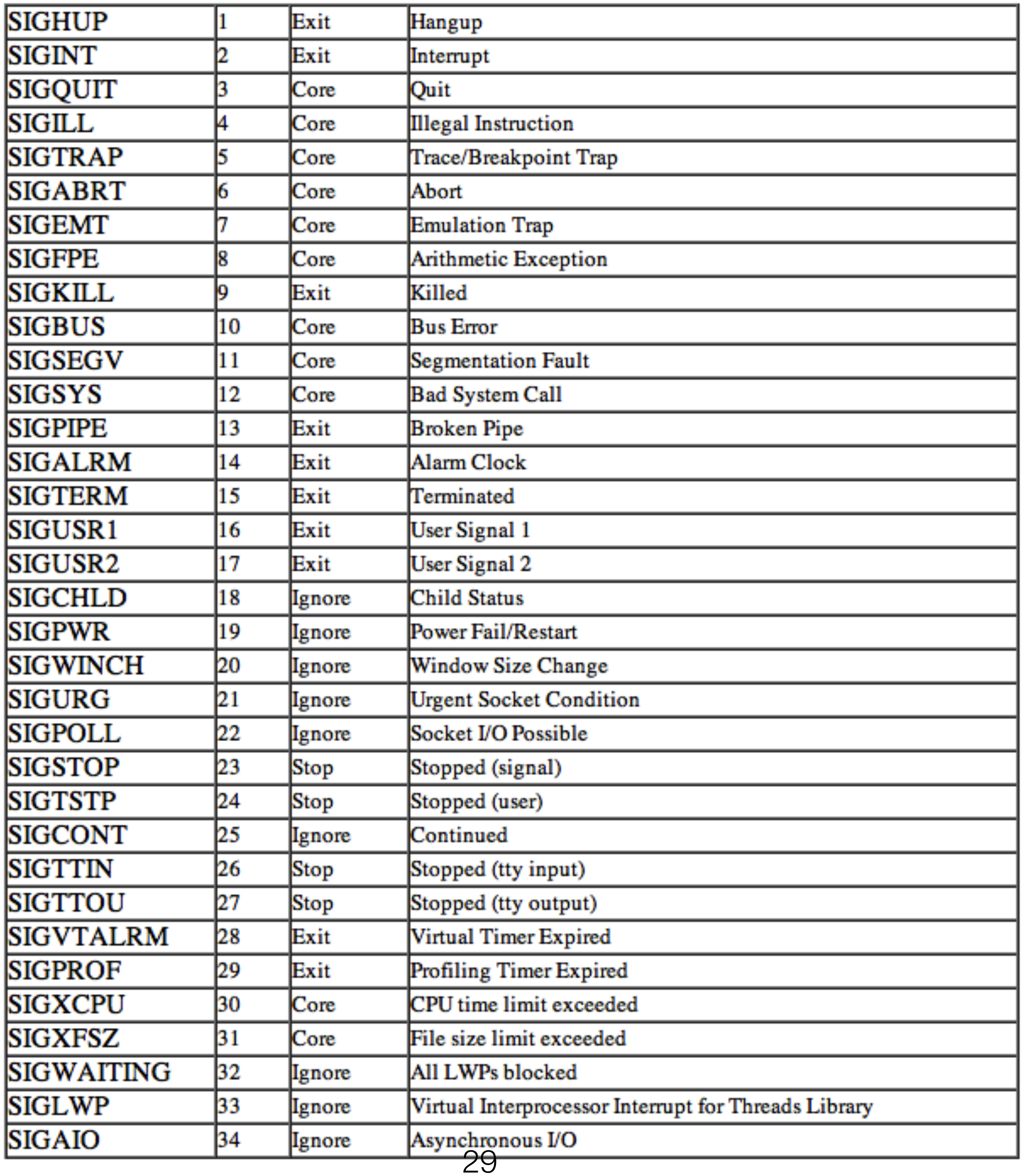
Using UNIX signals to reduce overhead : do not save the state at each iteration -- wait for the signal.

UNIX processes can receive 'signals' from the user, the OS, or another process



SIGHUP	1	Exit	Hangup
SIGINT	2	Exit	Interrupt
SIGQUIT	3	Core	Quit
SIGILL	4	Core	Illegal Instruction
SIGTRAP	5	Core	Trace/Breakpoint Trap
SIGABRT	6	Core	Abort
SIGEMT	7	Core	Emulation Trap
SIGFPE	8	Core	Arithmetic Exception
SIGKILL	9	Exit	Killed
SIGBUS	10	Core	Bus Error
SIGSEGV	11	Core	Segmentation Fault
SIGSYS	12	Core	Bad System Call
SIGPIPE	13	Exit	Broken Pipe
SIGALRM	14	Exit	Alarm Clock
SIGTERM	15	Exit	Terminated
SIGUSR1	16	Exit	User Signal 1
SIGUSR2	17	Exit	User Signal 2
SIGCHLD	18	Ignore	Child Status
SIGPWR	19	Ignore	Power Fail/Restart
SIGWINCH	20	Ignore	Window Size Change
SIGURG	21	Ignore	Urgent Socket Condition
SIGPOLL	22	Ignore	Socket I/O Possible
SIGSTOP	23	Stop	Stopped (signal)
SIGTSTP	24	Stop	Stopped (user)
SIGCONT	25	Ignore	Continued
SIGTTIN	26	Stop	Stopped (tty input)
SIGTTOU	27	Stop	Stopped (tty output)
SIGVTALRM	28	Exit	Virtual Timer Expired
SIGPROF	29	Exit	Profiling Timer Expired
SIGXCPU	30	Core	CPU time limit exceeded
SIGXFSZ	31	Core	File size limit exceeded
SIGWAITING	32	Ignore	All LWPs blocked
SIGLWP	33	Ignore	Virtual Interprocessor Interrupt for Threads Library
SIGAIO	34	Ignore	Asynchronous I/O

UNIX processes can receive 'signals' from the user, the OS, or another process



SIGHUP	1	Exit	Hangup
SIGINT	2	Exit	Interrupt
SIGQUIT	3	Core	Quit
SIGILL	4	Core	Illegal Instruction
SIGTRAP	5	Core	Trace/Breakpoint Trap
SIGABRT	6	Core	Abort
SIGEMT	7	Core	Emulation Trap
SIGFPE	8	Core	Arithmetic Exception
SIGKILL	9	Exit	Killed
SIGBUS	10	Core	Bus Error
SIGSEGV	11	Core	Segmentation Fault
SIGSYS	12	Core	Bad System Call
SIGPIPE	13	Exit	Broken Pipe
SIGALRM	14	Exit	Alarm Clock
SIGTERM	15	Exit	Terminated
SIGUSR1	16	Exit	User Signal 1
SIGUSR2	17	Exit	User Signal 2
SIGCHLD	18	Ignore	Child Status
SIGPWR	19	Ignore	Power Fail/Restart
SIGWINCH	20	Ignore	Window Size Change
SIGURG	21	Ignore	Urgent Socket Condition
SIGPOLL	22	Ignore	Socket I/O Possible
SIGSTOP	23	Stop	Stopped (signal)
SIGTSTP	24	Stop	Stopped (user)
SIGCONT	25	Ignore	Continued
SIGTTIN	26	Stop	Stopped (tty input)
SIGTTOU	27	Stop	Stopped (tty output)
SIGVTALRM	28	Exit	Virtual Timer Expired
SIGPROF	29	Exit	Profiling Timer Expired
SIGXCPU	30	Core	CPU time limit exceeded
SIGXFSZ	31	Core	File size limit exceeded
SIGWAITING	32	Ignore	All LWPs blocked
SIGLWP	33	Ignore	Virtual Interprocessor Interrupt for Threads Library
SIGAIO	34	Ignore	Asynchronous I/O

^C

^D

kill -9

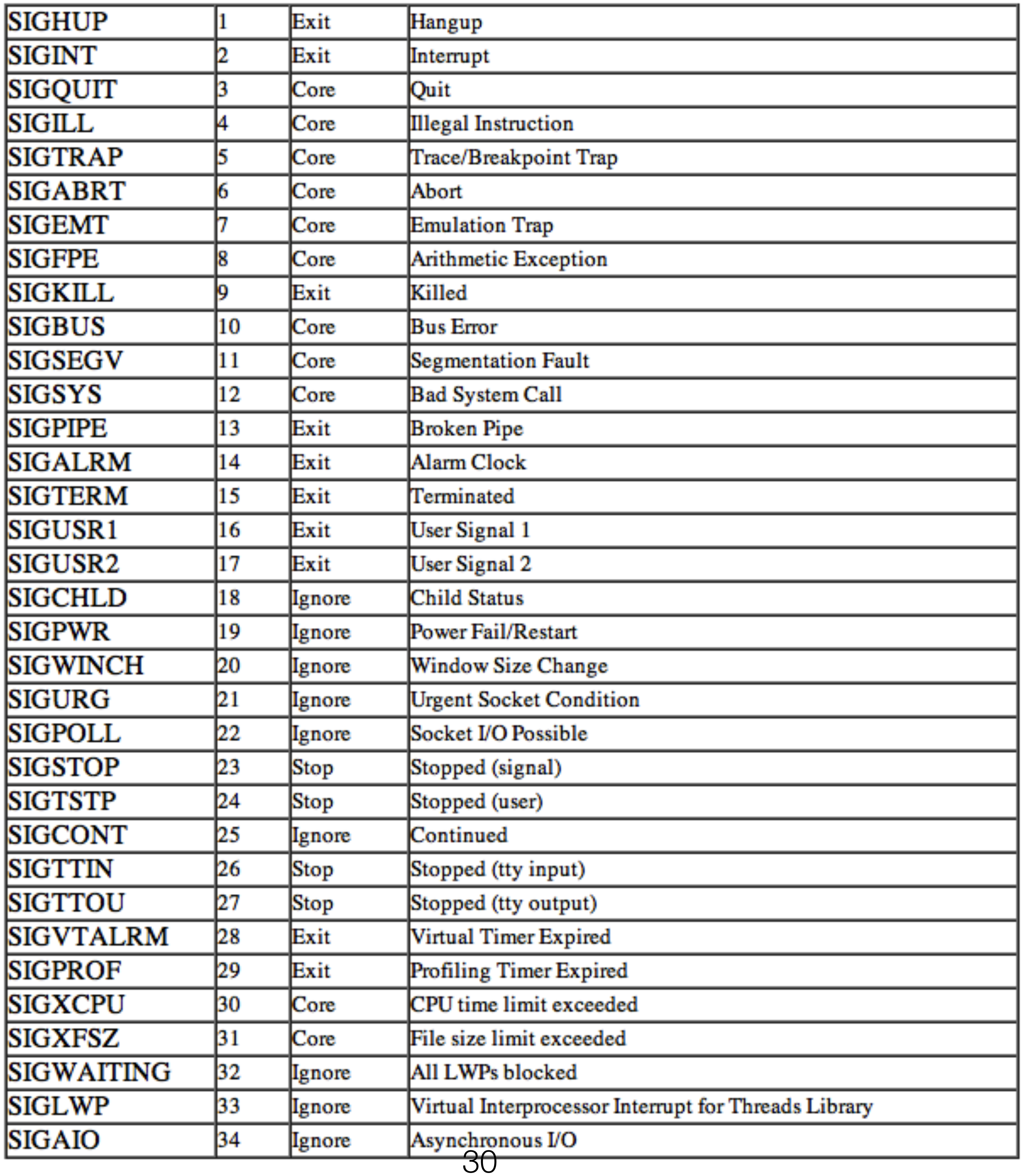
kill

^Z

fg, bg

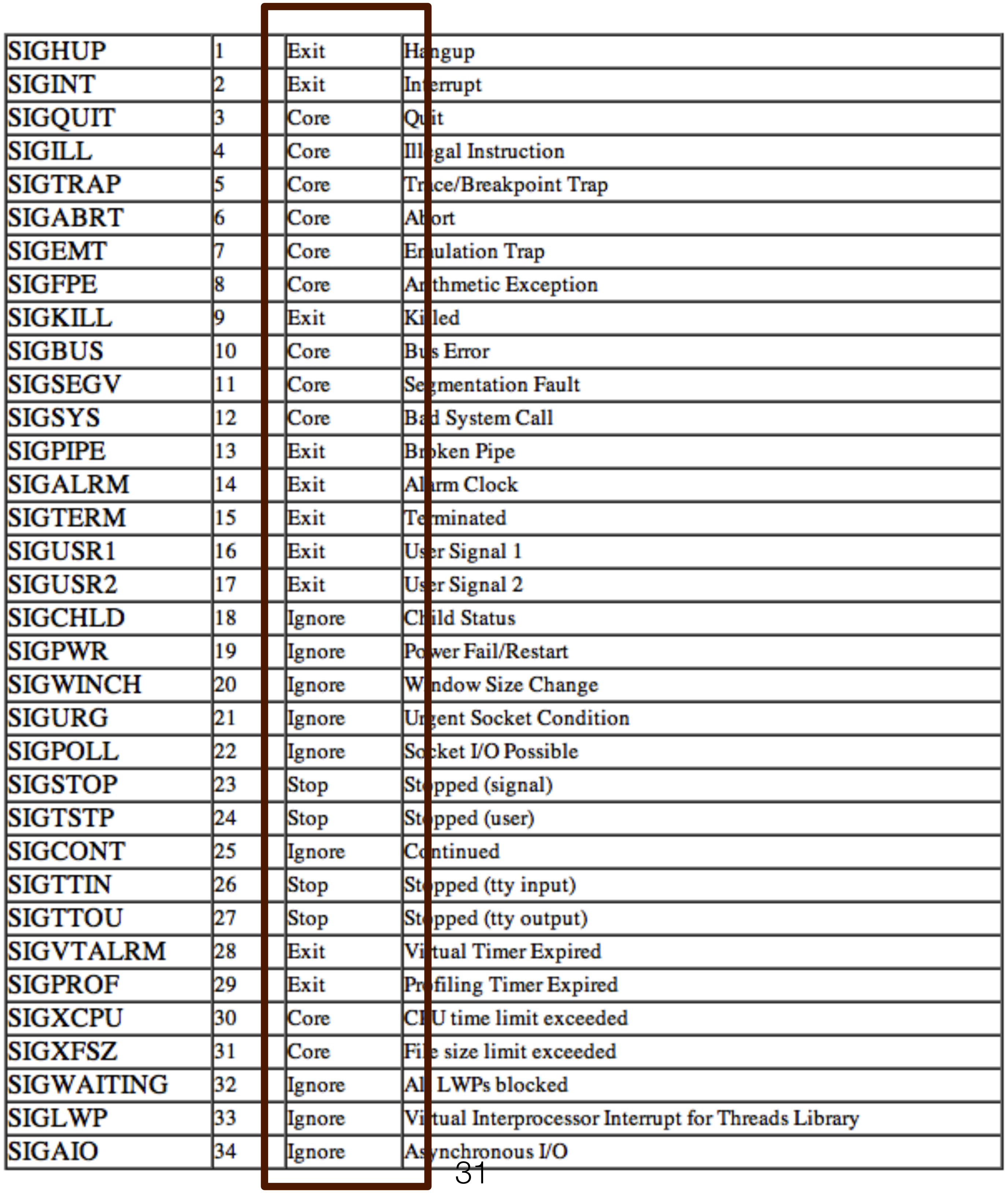
UNIX processes can receive 'signals' from the user, the OS, or another process

e.g.



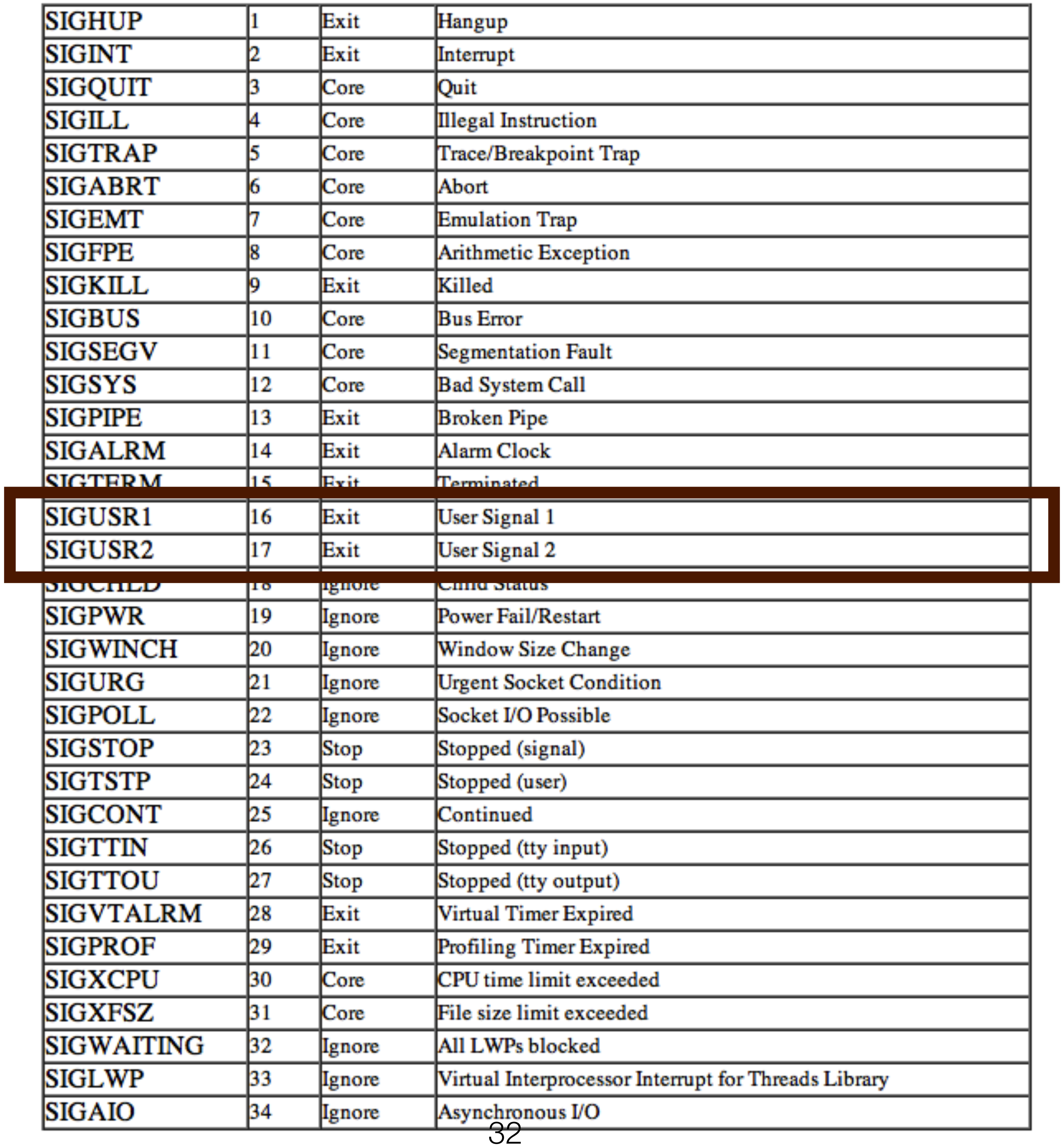
SIGHUP	1	Exit	Hangup
SIGINT	2	Exit	Interrupt
SIGQUIT	3	Core	Quit
SIGILL	4	Core	Illegal Instruction
SIGTRAP	5	Core	Trace/Breakpoint Trap
SIGABRT	6	Core	Abort
SIGEMT	7	Core	Emulation Trap
SIGFPE	8	Core	Arithmetic Exception
SIGKILL	9	Exit	Killed
SIGBUS	10	Core	Bus Error
SIGSEGV	11	Core	Segmentation Fault
SIGSYS	12	Core	Bad System Call
SIGPIPE	13	Exit	Broken Pipe
SIGALRM	14	Exit	Alarm Clock
SIGTERM	15	Exit	Terminated
SIGUSR1	16	Exit	User Signal 1
SIGUSR2	17	Exit	User Signal 2
SIGCHLD	18	Ignore	Child Status
SIGPWR	19	Ignore	Power Fail/Restart
SIGWINCH	20	Ignore	Window Size Change
SIGURG	21	Ignore	Urgent Socket Condition
SIGPOLL	22	Ignore	Socket I/O Possible
SIGSTOP	23	Stop	Stopped (signal)
SIGTSTP	24	Stop	Stopped (user)
SIGCONT	25	Ignore	Continued
SIGTTIN	26	Stop	Stopped (tty input)
SIGTTOU	27	Stop	Stopped (tty output)
SIGVTALRM	28	Exit	Virtual Timer Expired
SIGPROF	29	Exit	Profiling Timer Expired
SIGXCPU	30	Core	CPU time limit exceeded
SIGXFSZ	31	Core	File size limit exceeded
SIGWAITING	32	Ignore	All LWPs blocked
SIGLWP	33	Ignore	Virtual Interprocessor Interrupt for Threads Library
SIGAIO	34	Ignore	Asynchronous I/O

UNIX processes can receive 'signals' with an associated default action



SIGHUP	1	Exit	Hangup
SIGINT	2	Exit	Interrupt
SIGQUIT	3	Core	Quit
SIGILL	4	Core	Illegal Instruction
SIGTRAP	5	Core	Trace/Breakpoint Trap
SIGABRT	6	Core	Abort
SIGEMT	7	Core	Emulation Trap
SIGFPE	8	Core	Arithmetic Exception
SIGKILL	9	Exit	Killed
SIGBUS	10	Core	Bus Error
SIGSEGV	11	Core	Segmentation Fault
SIGSYS	12	Core	Bad System Call
SIGPIPE	13	Exit	Broken Pipe
SIGALRM	14	Exit	Alarm Clock
SIGTERM	15	Exit	Terminated
SIGUSR1	16	Exit	User Signal 1
SIGUSR2	17	Exit	User Signal 2
SIGCHLD	18	Ignore	Child Status
SIGPWR	19	Ignore	Power Fail/Restart
SIGWINCH	20	Ignore	Window Size Change
SIGURG	21	Ignore	Urgent Socket Condition
SIGPOLL	22	Ignore	Socket I/O Possible
SIGSTOP	23	Stop	Stopped (signal)
SIGTSTP	24	Stop	Stopped (user)
SIGCONT	25	Ignore	Continued
SIGTTIN	26	Stop	Stopped (tty input)
SIGTTOU	27	Stop	Stopped (tty output)
SIGVTALRM	28	Exit	Virtual Timer Expired
SIGPROF	29	Exit	Profiling Timer Expired
SIGXCPU	30	Core	CPU time limit exceeded
SIGXFSZ	31	Core	File size limit exceeded
SIGWAITING	32	Ignore	All LWPs blocked
SIGLWP	33	Ignore	Virtual Interprocessor Interrupt for Threads Library
SIGAIO	34	Ignore	Asynchronous I/O

UNIX processes can receive 'signals' with an associated default action



SIGHUP	1	Exit	Hangup
SIGINT	2	Exit	Interrupt
SIGQUIT	3	Core	Quit
SIGILL	4	Core	Illegal Instruction
SIGTRAP	5	Core	Trace/Breakpoint Trap
SIGABRT	6	Core	Abort
SIGEMT	7	Core	Emulation Trap
SIGFPE	8	Core	Arithmetic Exception
SIGKILL	9	Exit	Killed
SIGBUS	10	Core	Bus Error
SIGSEGV	11	Core	Segmentation Fault
SIGSYS	12	Core	Bad System Call
SIGPIPE	13	Exit	Broken Pipe
SIGALRM	14	Exit	Alarm Clock
SIGTERM	15	Exit	Terminated
SIGUSR1	16	Exit	User Signal 1
SIGUSR2	17	Exit	User Signal 2
SIGCHLD	18	Ignore	Child Status
SIGPWR	19	Ignore	Power Fail/Restart
SIGWINCH	20	Ignore	Window Size Change
SIGURG	21	Ignore	Urgent Socket Condition
SIGPOLL	22	Ignore	Socket I/O Possible
SIGSTOP	23	Stop	Stopped (signal)
SIGTSTP	24	Stop	Stopped (user)
SIGCONT	25	Ignore	Continued
SIGTTIN	26	Stop	Stopped (tty input)
SIGTTOU	27	Stop	Stopped (tty output)
SIGVTALRM	28	Exit	Virtual Timer Expired
SIGPROF	29	Exit	Profiling Timer Expired
SIGXCPU	30	Core	CPU time limit exceeded
SIGXFSZ	31	Core	File size limit exceeded
SIGWAITING	32	Ignore	All LWPs blocked
SIGLWP	33	Ignore	Virtual Interprocessor Interrupt for Threads Library
SIGAIO	34	Ignore	Asynchronous I/O

Demo #2

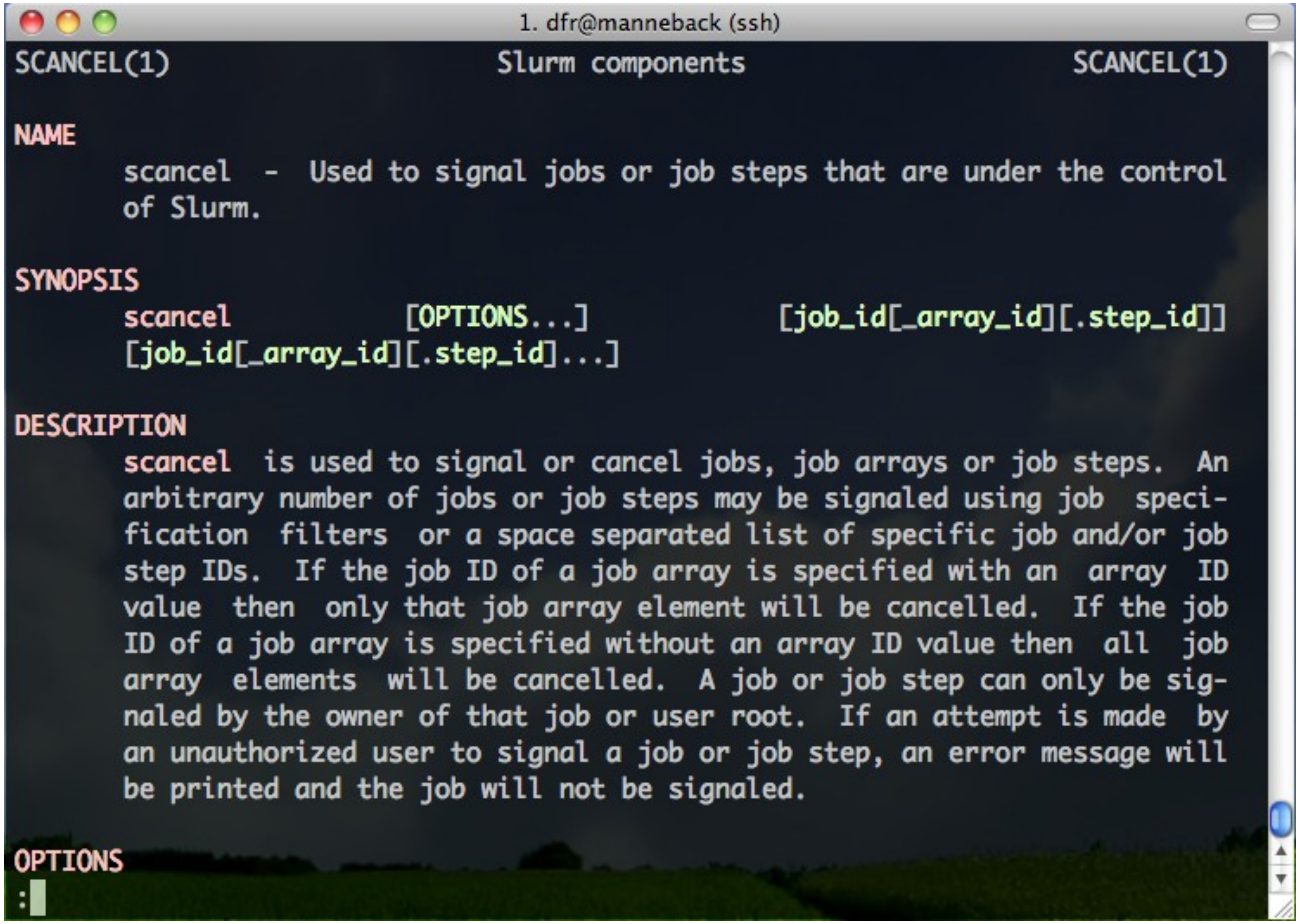
count-signal.py

Catch control-C to save state

3

Use Slurm signaling abilities to manage checkpoint-able software in Slurm scripts on the clusters.

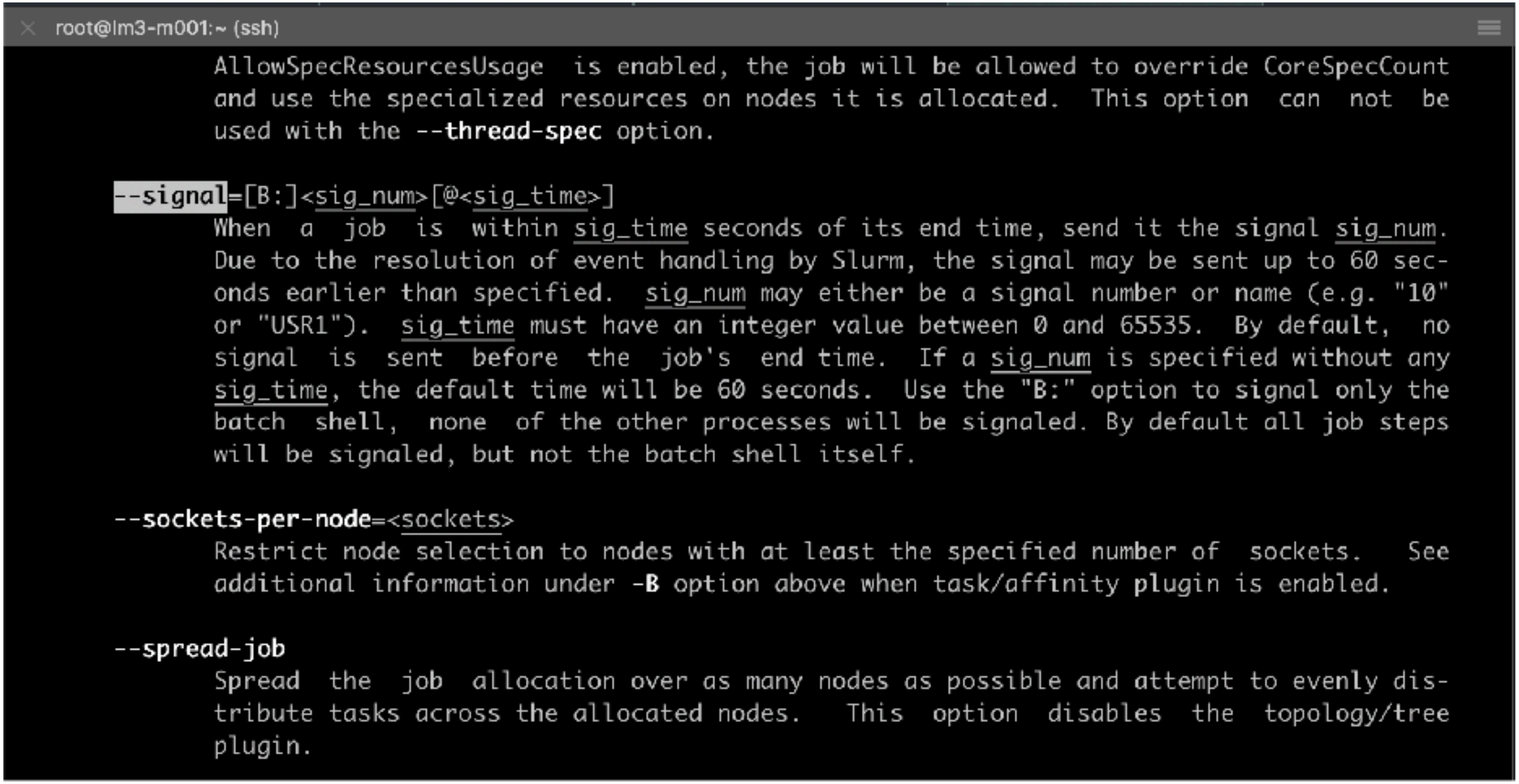
scancel is used to send signals to jobs



```
SCANCEL(1)                 Slurm components                 SCANCEL(1)

NAME
       scancel  -  Used to signal jobs or job steps that are under the control
       of Slurm.

SYNOPSIS
       scancel            [OPTIONS...]            [job_id[_array_id][.step_id]]
       [job_id[_array_id][.step_id]...]

DESCRIPTION
       scancel  is used to signal or cancel jobs, job arrays or job steps.  An
       arbitrary number of jobs or job steps may be signaled using job  speci-
       fication  filters  or a space separated list of specific job and/or job
       step IDs.  If the job ID of a job array is specified with an  array  ID
       value  then  only that job array element will be cancelled.  If the job
       ID of a job array is specified without an array ID value then  all  job
       array  elements  will be cancelled.  A job or job step can only be sig-
       naled by the owner of that job or user root.  If an attempt is made  by
       an unauthorized user to signal a job or job step, an error message will
       be printed and the job will not be signaled.

OPTIONS
:
```

scancel -s SIGINT JOBID

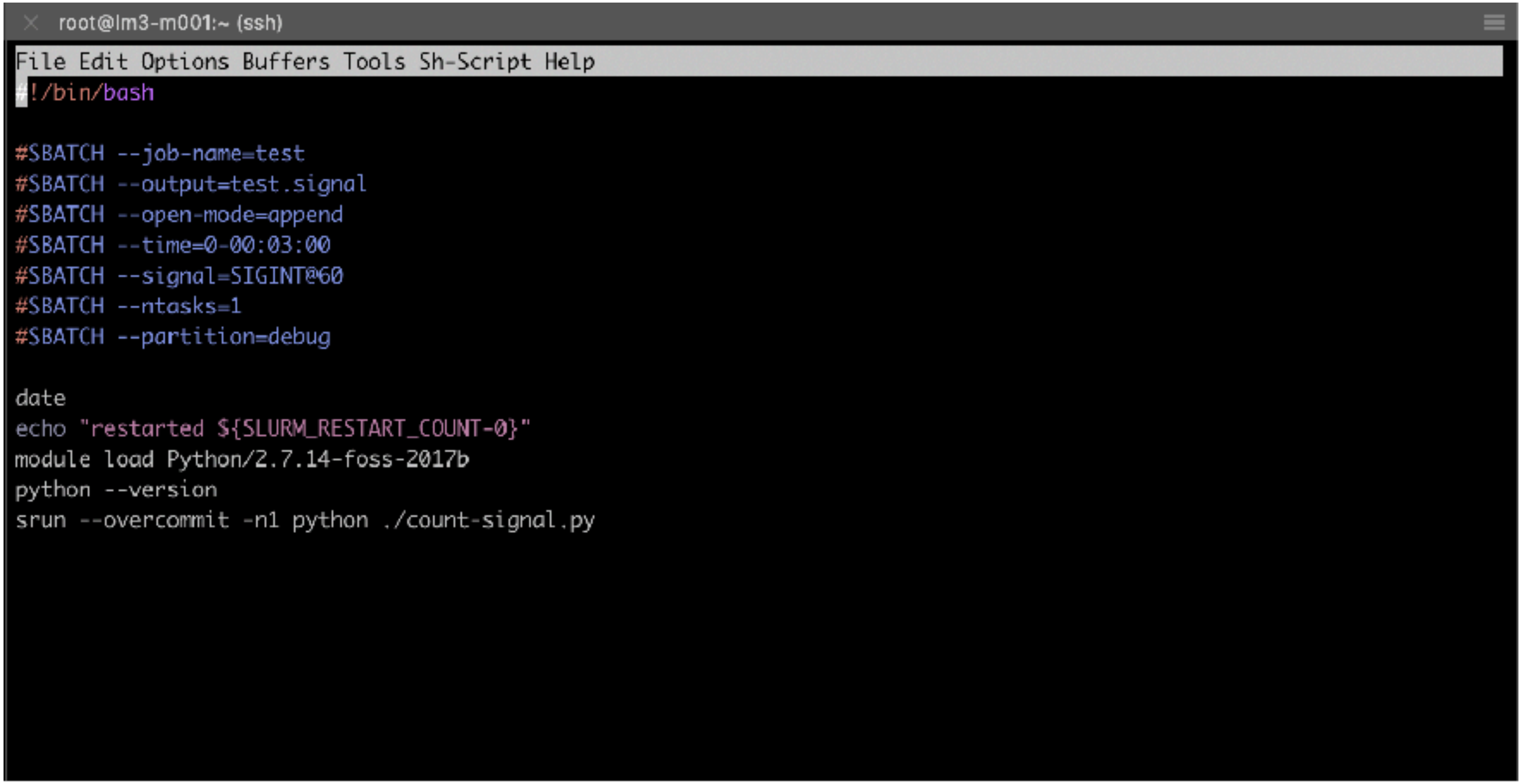
--signal to have Slurm send signals automatically before the end of the allocation



```
              AllowSpecResourcesUsage  is enabled, the job will be allowed to override CoreSpecCount
              and use the specialized resources on nodes it is allocated.  This option  can  not  be
              used with the --thread-spec option.

       --signal=[B:]<sig_num>[@<sig_time>]
              When  a  job  is  within sig_time seconds of its end time, send it the signal sig_num.
              Due to the resolution of event handling by Slurm, the signal may be sent up to 60 sec-
              onds earlier than specified.  sig_num may either be a signal number or name (e.g. "10"
              or "USR1").  sig_time must have an integer value between 0 and 65535.  By default,  no
              signal  is  sent  before  the  job's  end time.  If a sig_num is specified without any
              sig_time, the default time will be 60 seconds.  Use the "B:" option to signal only the
              batch  shell,  none  of the other processes will be signaled. By default all job steps
              will be signaled, but not the batch shell itself.

       --sockets-per-node=<sockets>
              Restrict node selection to nodes with at least the specified number of  sockets.   See
              additional information under -B option above when task/affinity plugin is enabled.

       --spread-job
              Spread  the  job  allocation over as many nodes as possible and attempt to evenly dis-
              tribute tasks across the allocated nodes.   This  option  disables  the  topology/tree
              plugin.
```

--signal=B:SIGINT send signal to the bash script
--signal=SIGINT send signal to the srun command

Note the --open-mode=append



```
#!/bin/bash

#SBATCH --job-name=test
#SBATCH --output=test.signal
#SBATCH --open-mode=append
#SBATCH --time=0-00:03:00
#SBATCH --signal=SIGINT@60
#SBATCH --ntasks=1
#SBATCH --partition=debug

date
echo "restarted ${SLURM_RESTART_COUNT-0}"
module load Python/2.7.14-foss-2017b
python --version
srun --overcommit -n1 python ./count-signal.py
```

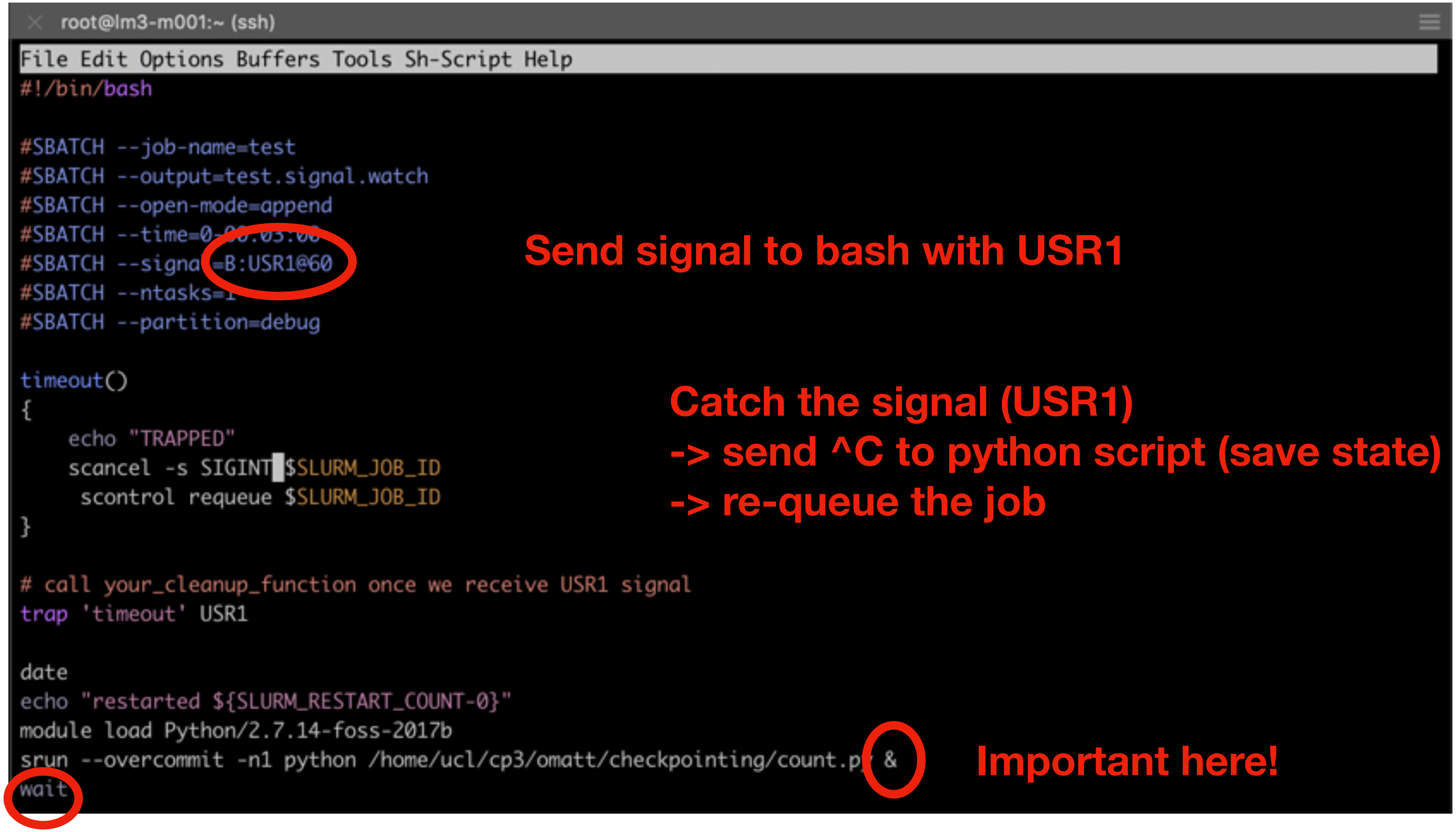
Note that we need the srun here

Demo #3

submit-signal.sh
python: Catch control-C to save state
Slurm send control-C between 1 and 2 minutes

submit-signal2.sh
python: Catch control-C to save state
Slurm send control-C between 1 and 2 minutes
Automatic re-queuing

Adding requeuing automatically



```
#!/bin/bash

#SBATCH --job-name=test
#SBATCH --output=test.signal.watch
#SBATCH --open-mode=append
#SBATCH --time=0-00:03:00
#SBATCH --signal=B:USR1@60
#SBATCH --ntasks=1
#SBATCH --partition=debug

timeout()
{
    echo "TRAPPED"
    scancel -s SIGINT $SLURM_JOB_ID
     scontrol requeue $SLURM_JOB_ID
}

# call your_cleanup_function once we receive USR1 signal
trap 'timeout' USR1

date
echo "restarted ${SLURM_RESTART_COUNT-0}"
module load Python/2.7.14-foss-2017b
srun --overcommit -n1 python /home/ucl/cp3/omatt/checkpointing/count.py &
wait
```

Send signal to bash with USR1

Catch the signal (USR1)
-> send ^C to python script (save state)
-> re-queue the job

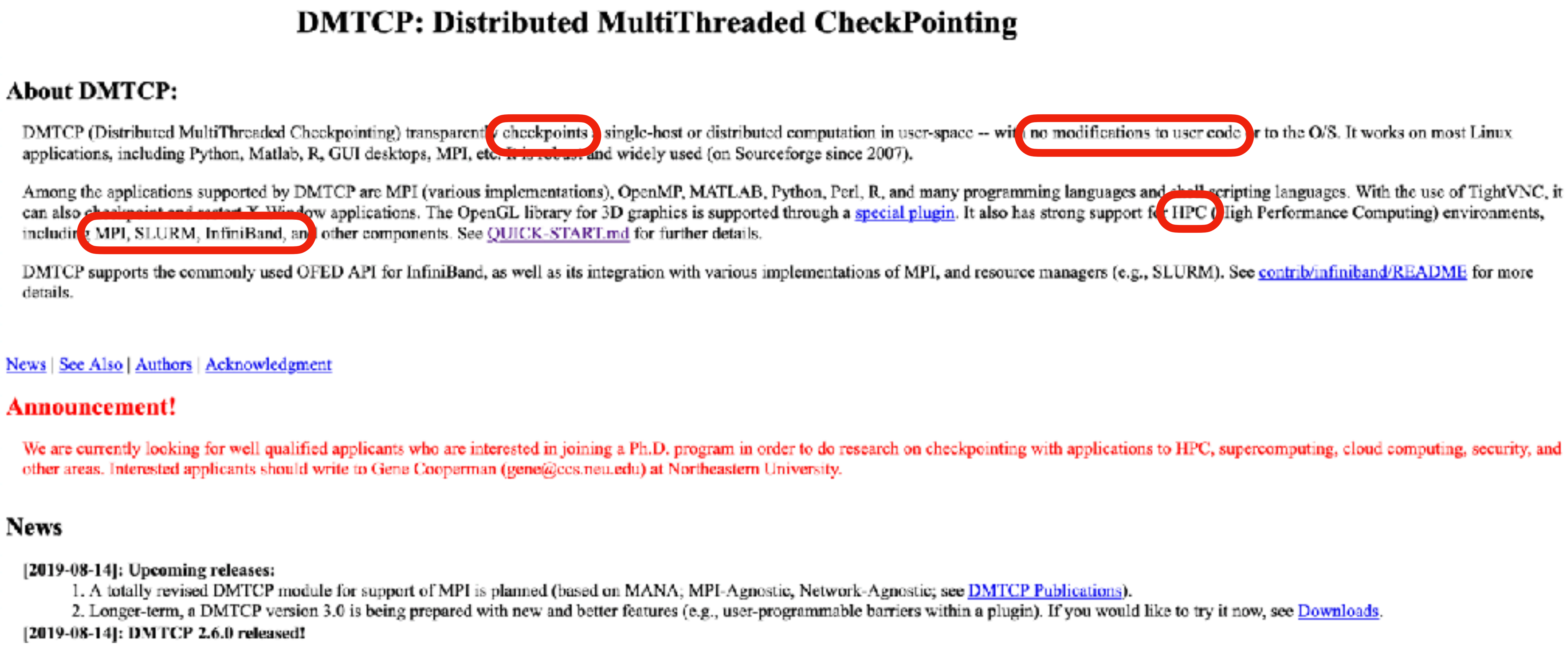
Important here!

Demo #4

slurm-signal-3.sh

Slurm send USR1 between 1 and 2 minutes
Bash catch the message send Ctrl-c to python
python: Catch control-C to save state
Automatic resubmission

4 Making non restartable software restartable with DMTCP



DMTCP: Distributed MultiThreaded CheckPointing

About DMTCP:

DMTCP (Distributed MultiThreaded Checkpointing) transparently checkpoints a single-host or distributed computation in user-space -- with no modifications to user code or to the O/S. It works on most Linux applications, including Python, Matlab, R, GUI desktops, MPI, etc. It is robust and widely used (on Sourceforge since 2007).

Among the applications supported by DMTCP are MPI (various implementations), OpenMP, MATLAB, Python, Perl, R, and many programming languages and shell scripting languages. With the use of TightVNC, it can also checkpoint and restart X Window applications. The OpenGL library for 3D graphics is supported through a special plugin. It also has strong support for HPC (High Performance Computing) environments, including MPI, SLURM, InfiniBand, and other components. See QUICK-START.md for further details.

DMTCP supports the commonly used OFED API for InfiniBand, as well as its integration with various implementations of MPI, and resource managers (e.g., SLURM). See contrib/infiniband/README for more details.

News | See Also | Authors | Acknowledgment

Announcement!

We are currently looking for well qualified applicants who are interested in joining a Ph.D. program in order to do research on checkpointing with applications to HPC, supercomputing, cloud computing, security, and other areas. Interested applicants should write to Gene Cooperman (gene@ccs.neu.edu) at Northeastern University.

News

[2019-08-14]: Upcoming releases:

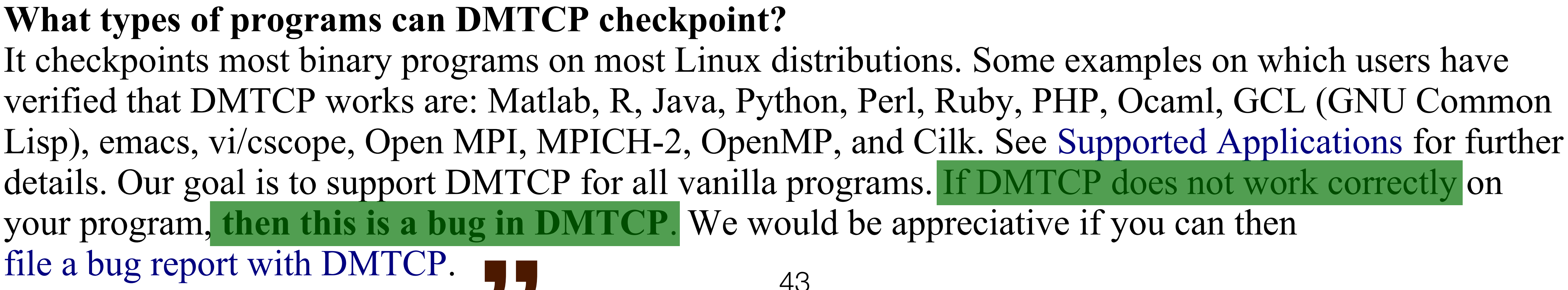
1. A totally revised DMTCP module for support of MPI is planned (based on MANA; MPI-Agnostic, Network-Agnostic; see DMTCP Publications).
2. Longer-term, a DMTCP version 3.0 is being prepared with new and better features (e.g., user-programmable barriers within a plugin). If you would like to try it now, see Downloads.

[2019-08-14]: DMTCP 2.6.0 released!

Advertised Features

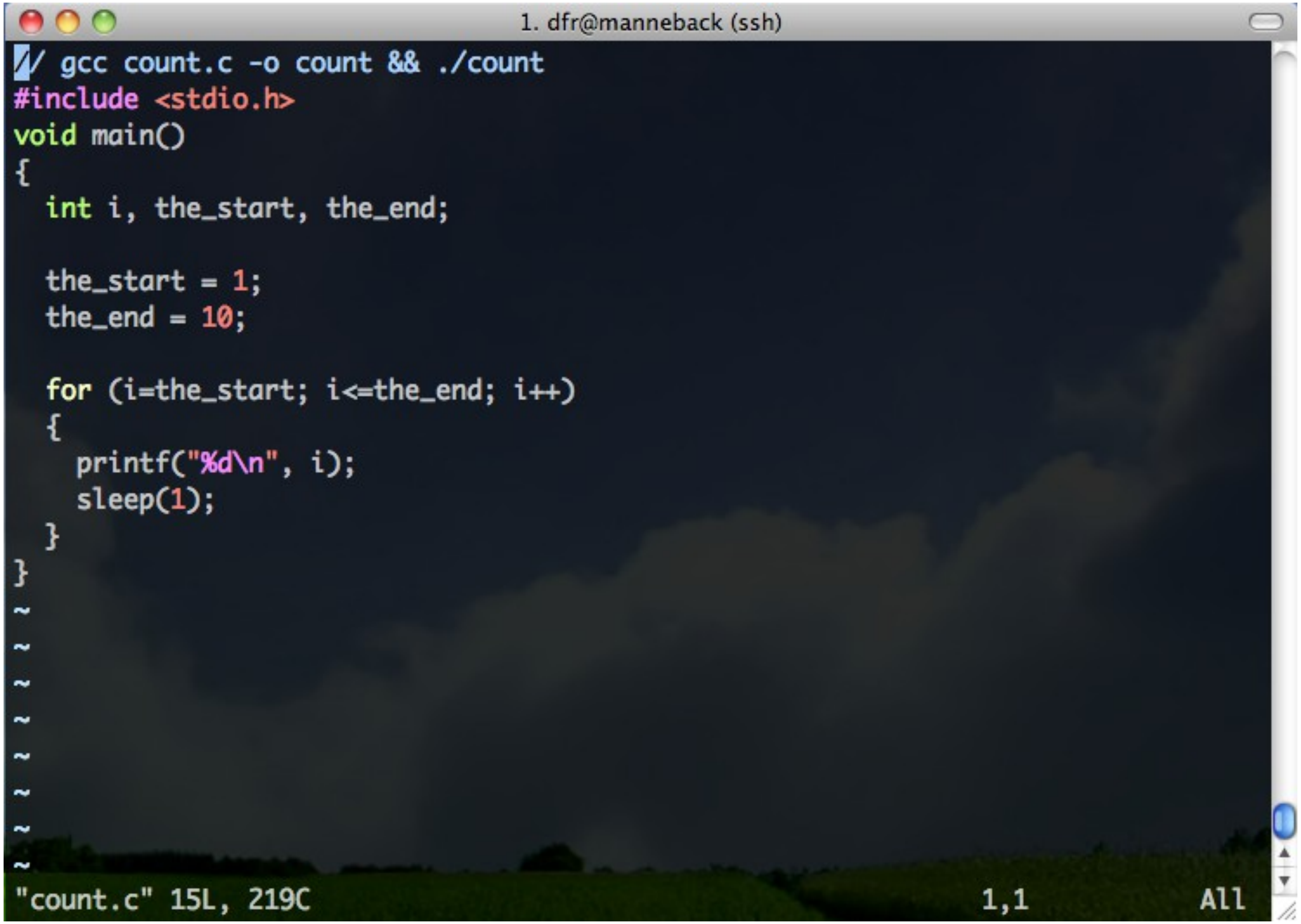
- Distributed Multi-Threaded CheckPointing
- Works with Linux Kernel 2.6.9 and later
- Supports sequential and multi-threaded computations across single/multiple hosts
- Entirely in user space (no kernel modules or root privilege)
- Transparent (no recompiling, no re-linking)
- Written at Northeastern U. and MIT and under active development for 5+ years
- LGPL'd and freely available
- No remote I/O
- Supports threads, mutexes/semaphoes, forks, shared memory, exec, and many more

From their FAQ:



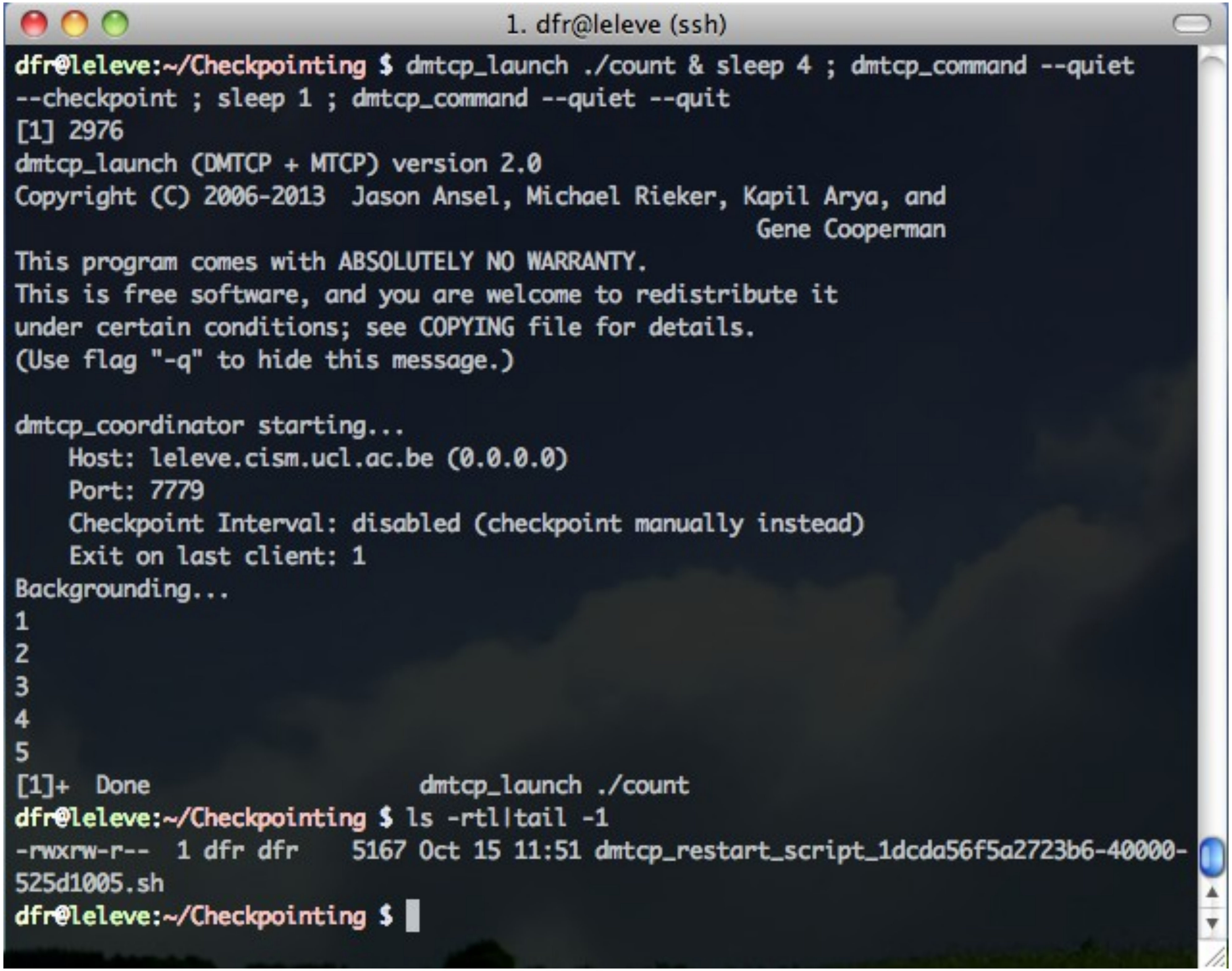
> **What types of programs can DMTCP checkpoint?**
> It checkpoints most binary programs on most Linux distributions. Some examples on which users have verified that DMTCP works are: Matlab, R, Java, Python, Perl, Ruby, PHP, Ocaml, GCL (GNU Common Lisp), emacs, vi/cscope, Open MPI, MPICH-2, OpenMP, and Cilk. See Supported Applications for further details. Our goal is to support DMTCP for all vanilla programs. If DMTCP does not work correctly on your program, **then this is a bug in DMTCP**. We would be appreciative if you can then file a bug report with DMTCP.

Imagine a non-checkpointable program



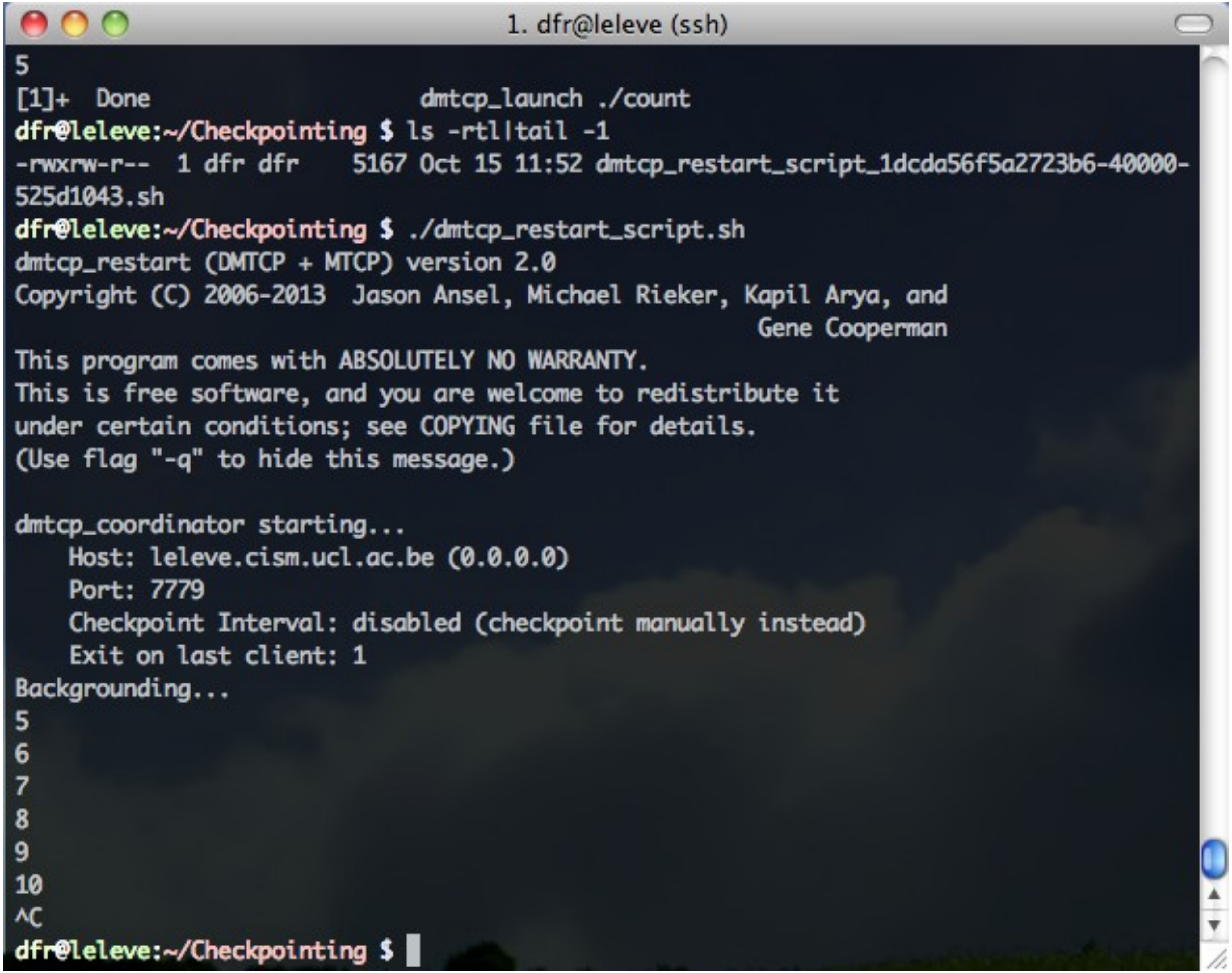
```
// gcc count.c -o count && ./count
#include <stdio.h>
void main()
{
  int i, the_start, the_end;

  the_start = 1;
  the_end = 10;

  for (i=the_start; i<=the_end; i++)
  {
    printf("%d\n", i);
    sleep(1);
  }
}
~
~
~
~
~
~
~
~
"count.c" 15L, 219C                                          1,1           All
```

Run with dmtcp_launch (runs monitoring daemon if necessary)



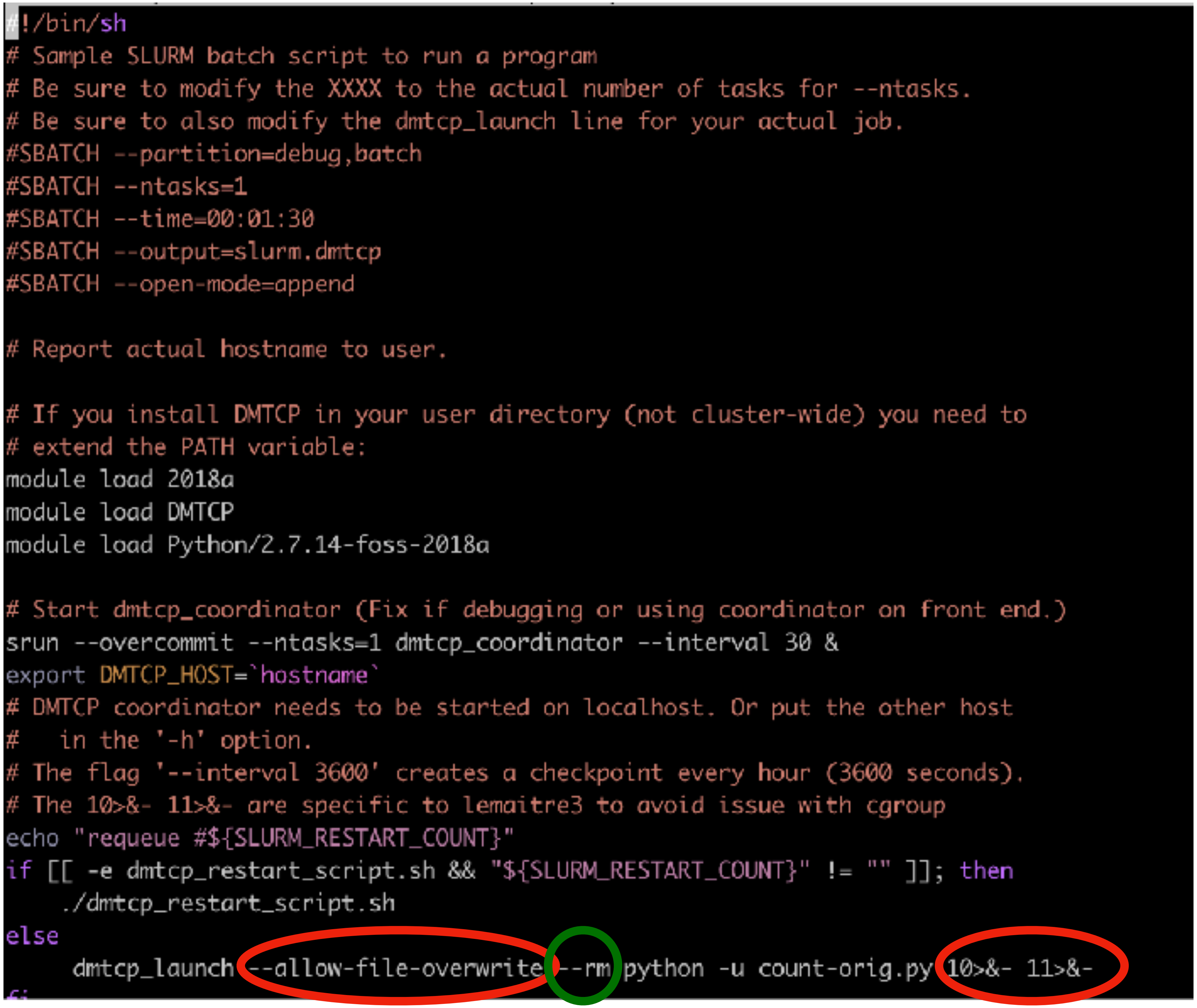
```
dfr@leleve:~/Checkpointing $ dmtcp_launch ./count & sleep 4 ; dmtcp_command --quiet
--checkpoint ; sleep 1 ; dmtcp_command --quiet --quit
[1] 2976
dmtcp_launch (DMTCP + MTCP) version 2.0
Copyright (C) 2006-2013  Jason Ansel, Michael Rieker, Kapil Arya, and
                                                       Gene Cooperman
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions; see COPYING file for details.
(Use flag "-q" to hide this message.)

dmtcp_coordinator starting...
    Host: leleve.cism.ucl.ac.be (0.0.0.0)
    Port: 7779
    Checkpoint Interval: disabled (checkpoint manually instead)
    Exit on last client: 1
Backgrounding...
1
2
3
4
5
[1]+  Done                    dmtcp_launch ./count
dfr@leleve:~/Checkpointing $ ls -rtl|tail -1
-rwxrw-r--  1 dfr dfr    5167 Oct 15 11:51 dmtcp_restart_script_1dcda56f5a2723b6-40000-
525d1005.sh
dfr@leleve:~/Checkpointing $ 
```

Restart with dmtcp_restart_script.sh



```
5
[1]+  Done                    dmtcp_launch ./count
dfr@leleve:~/Checkpointing $ ls -rtl|tail -1
-rwxrw-r--  1 dfr dfr    5167 Oct 15 11:52 dmtcp_restart_script_1dcda56f5a2723b6-40000-
525d1043.sh
dfr@leleve:~/Checkpointing $ ./dmtcp_restart_script.sh
dmtcp_restart (DMTCP + MTCP) version 2.0
Copyright (C) 2006-2013  Jason Ansel, Michael Rieker, Kapil Arya, and
                                                       Gene Cooperman
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions; see COPYING file for details.
(Use flag "-q" to hide this message.)

dmtcp_coordinator starting...
    Host: leleve.cism.ucl.ac.be (0.0.0.0)
    Port: 7779
    Checkpoint Interval: disabled (checkpoint manually instead)
    Exit on last client: 1
Backgrounding...
5
6
7
8
9
10
^C
dfr@leleve:~/Checkpointing $ 
```

Launch the coordinator and the program with automatic checkpointing every 30 seconds



```
#!/bin/sh
# Sample SLURM batch script to run a program
# Be sure to modify the XXXX to the actual number of tasks for --ntasks.
# Be sure to also modify the dmtcp_launch line for your actual job.
#SBATCH --partition=debug,batch
#SBATCH --ntasks=1
#SBATCH --time=00:01:30
#SBATCH --output=slurm.dmtcp
#SBATCH --open-mode=append

# Report actual hostname to user.

# If you install DMTCP in your user directory (not cluster-wide) you need to
# extend the PATH variable:
module load 2018a
module load DMTCP
module load Python/2.7.14-foss-2018a

# Start dmtcp_coordinator (Fix if debugging or using coordinator on front end.)
srun --overcommit --ntasks=1 dmtcp_coordinator --interval 30 &
export DMTCP_HOST=`hostname`
# DMTCP coordinator needs to be started on localhost. Or put the other host
#   in the '-h' option.
# The flag '--interval 3600' creates a checkpoint every hour (3600 seconds).
# The 10>&- 11>&- are specific to lemaitre3 to avoid issue with cgroup
echo "requeue #${SLURM_RESTART_COUNT}"
if [[ -e dmtcp_restart_script.sh && "${SLURM_RESTART_COUNT}" != "" ]]; then
    ./dmtcp_restart_script.sh
else
    dmtcp_launch --allow-file-overwrite --rm python -u count-orig.py 10>&- 11>&-
fi
```

start coordinator

Check if need restart

Normal job with decorator

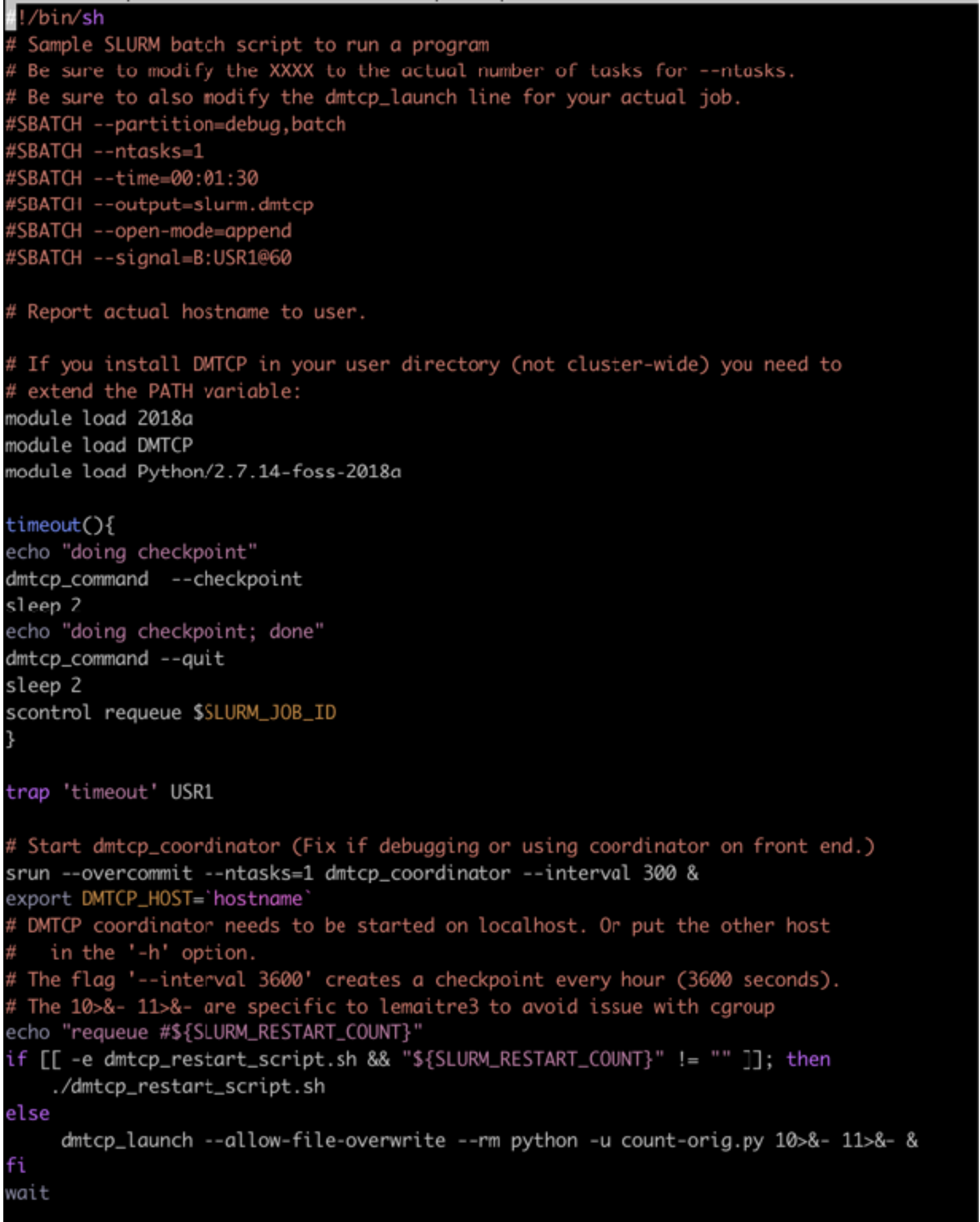
Resubmit

Lemaitre3 specific!

Let’s combine everything

Use DMTCP and add an additional checkpoint
Just before wall time

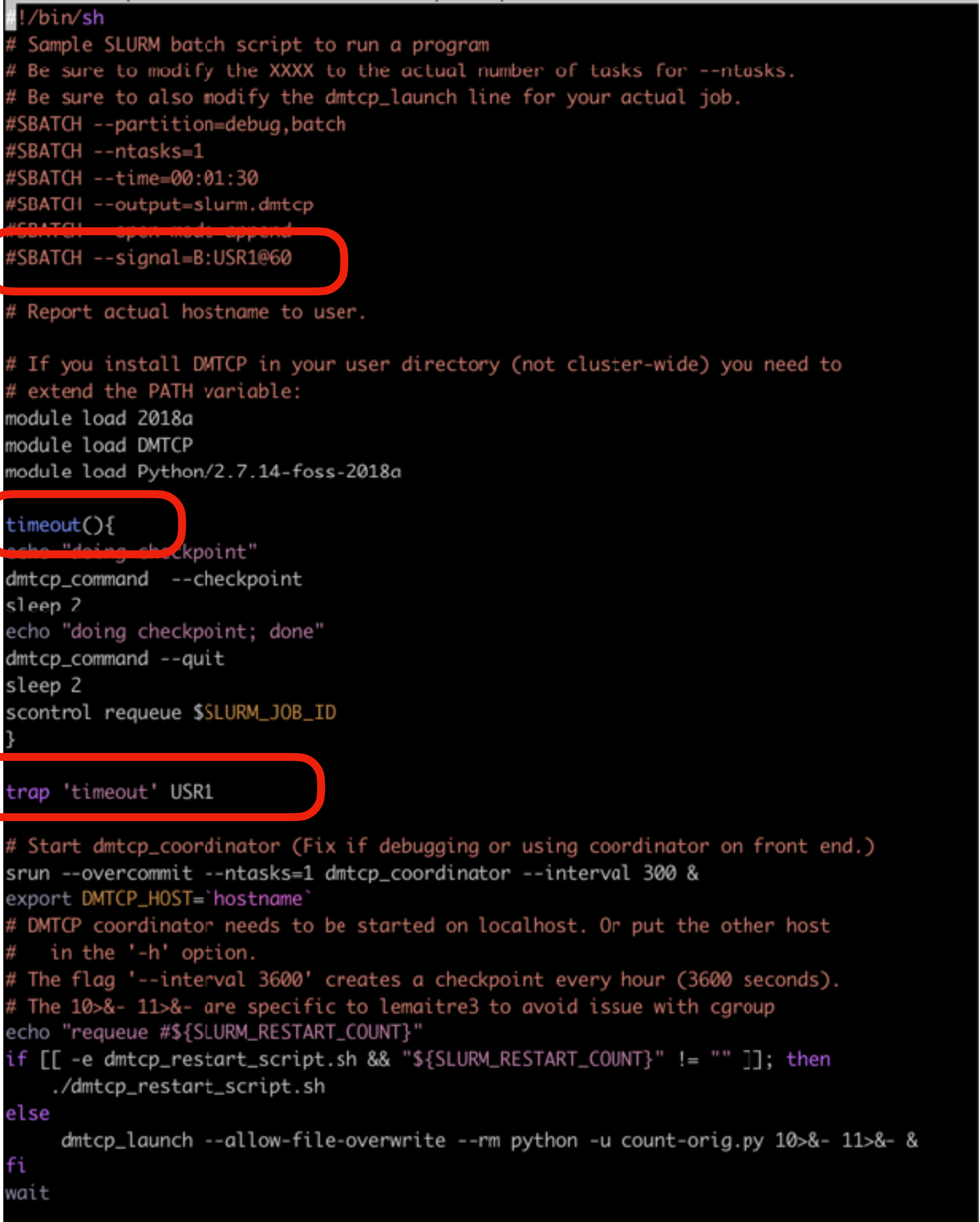
Solution



```
#!/bin/sh
# Sample SLURM batch script to run a program
# Be sure to modify the XXXX to the actual number of tasks for --ntasks.
# Be sure to also modify the dmtcp_launch line for your actual job.
#SBATCH --partition=debug,batch
#SBATCH --ntasks=1
#SBATCH --time=00:01:30
#SBATCH --output=slurm.dmtcp
#SBATCH --open-mode=append
#SBATCH --signal=B:USR1@60

# Report actual hostname to user.

# If you install DMTCP in your user directory (not cluster-wide) you need to
# extend the PATH variable:
module load 2018a
module load DMTCP
module load Python/2.7.14-foss-2018a

timeout(){
echo "doing checkpoint"
dmtcp_command  --checkpoint
sleep 2
echo "doing checkpoint; done"
dmtcp_command --quit
sleep 2
scontrol requeue $SLURM_JOB_ID
}

trap 'timeout' USR1

# Start dmtcp_coordinator (Fix if debugging or using coordinator on front end.)
srun --overcommit --ntasks=1 dmtcp_coordinator --interval 300 &
export DMTCP_HOST=`hostname`
# DMTCP coordinator needs to be started on localhost. Or put the other host
#   in the '-h' option.
# The flag '--interval 3600' creates a checkpoint every hour (3600 seconds).
# The 10>&- 11>&- are specific to lemaitre3 to avoid issue with cgroup
echo "requeue #${SLURM_RESTART_COUNT}"
if [[ -e dmtcp_restart_script.sh && "${SLURM_RESTART_COUNT}" != "" ]]; then
    ./dmtcp_restart_script.sh
else
     dmtcp_launch --allow-file-overwrite --rm python -u count-orig.py 10>&- 11>&- &
fi
wait
```

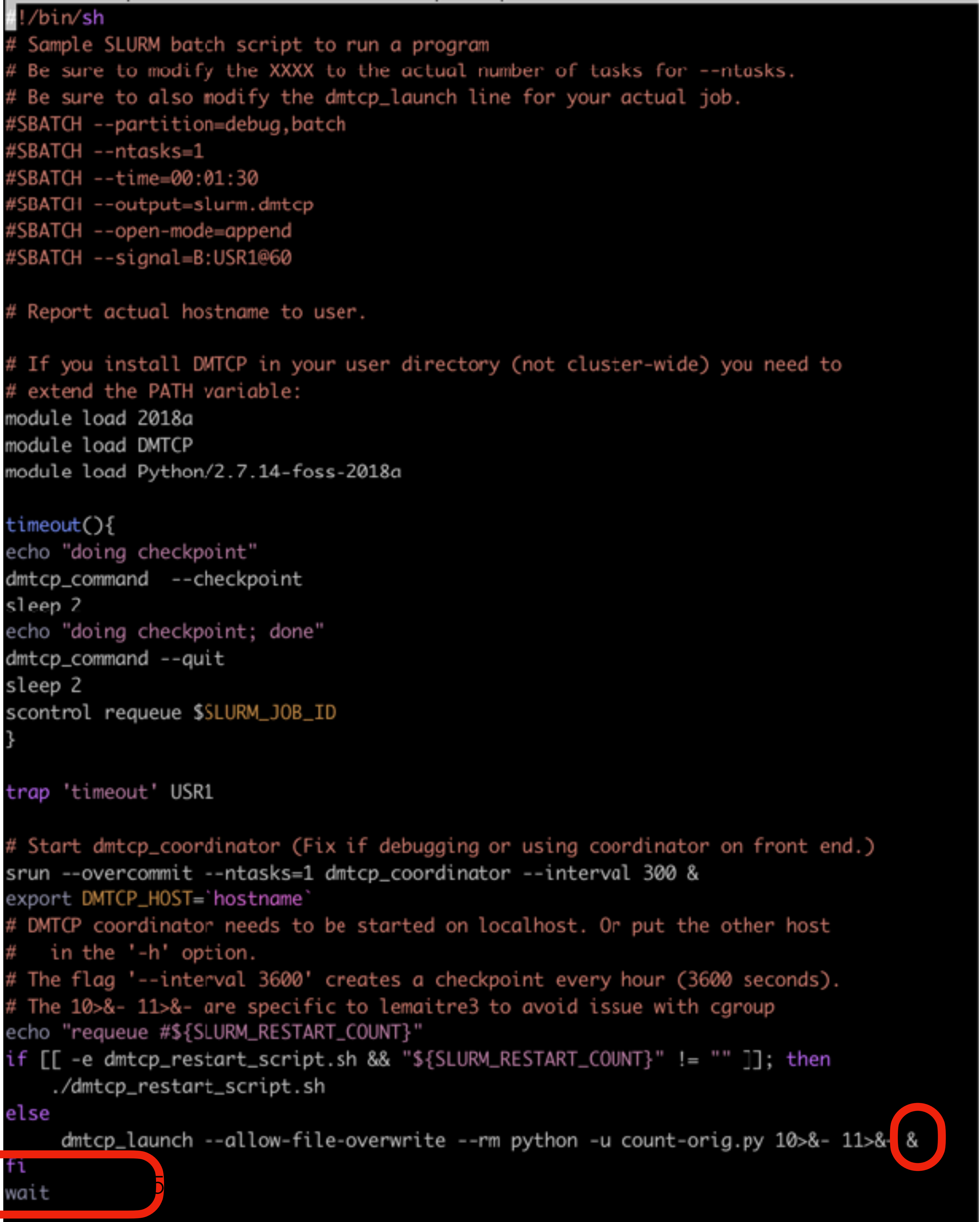
Solution

Setup signal



```
#!/bin/sh
# Sample SLURM batch script to run a program
# Be sure to modify the XXXX to the actual number of tasks for --ntasks.
# Be sure to also modify the dmtcp_launch line for your actual job.
#SBATCH --partition=debug,batch
#SBATCH --ntasks=1
#SBATCH --time=00:01:30
#SBATCH --output=slurm.dmtcp
#SBATCH --open-mode=append
#SBATCH --signal=B:USR1@60

# Report actual hostname to user.

# If you install DMTCP in your user directory (not cluster-wide) you need to
# extend the PATH variable:
module load 2018a
module load DMTCP
module load Python/2.7.14-foss-2018a

timeout(){
echo "doing checkpoint"
dmtcp_command  --checkpoint
sleep 2
echo "doing checkpoint; done"
dmtcp_command --quit
sleep 2
scontrol requeue $SLURM_JOB_ID
}

trap 'timeout' USR1

# Start dmtcp_coordinator (Fix if debugging or using coordinator on front end.)
srun --overcommit --ntasks=1 dmtcp_coordinator --interval 300 &
export DMTCP_HOST=`hostname`
# DMTCP coordinator needs to be started on localhost. Or put the other host
#   in the '-h' option.
# The flag '--interval 3600' creates a checkpoint every hour (3600 seconds).
# The 10>&- 11>&- are specific to lemaitre3 to avoid issue with cgroup
echo "requeue #${SLURM_RESTART_COUNT}"
if [[ -e dmtcp_restart_script.sh && "${SLURM_RESTART_COUNT}" != "" ]]; then
    ./dmtcp_restart_script.sh
else
     dmtcp_launch --allow-file-overwrite --rm python -u count-orig.py 10>&- 11>&- &
fi
wait
```

Solution

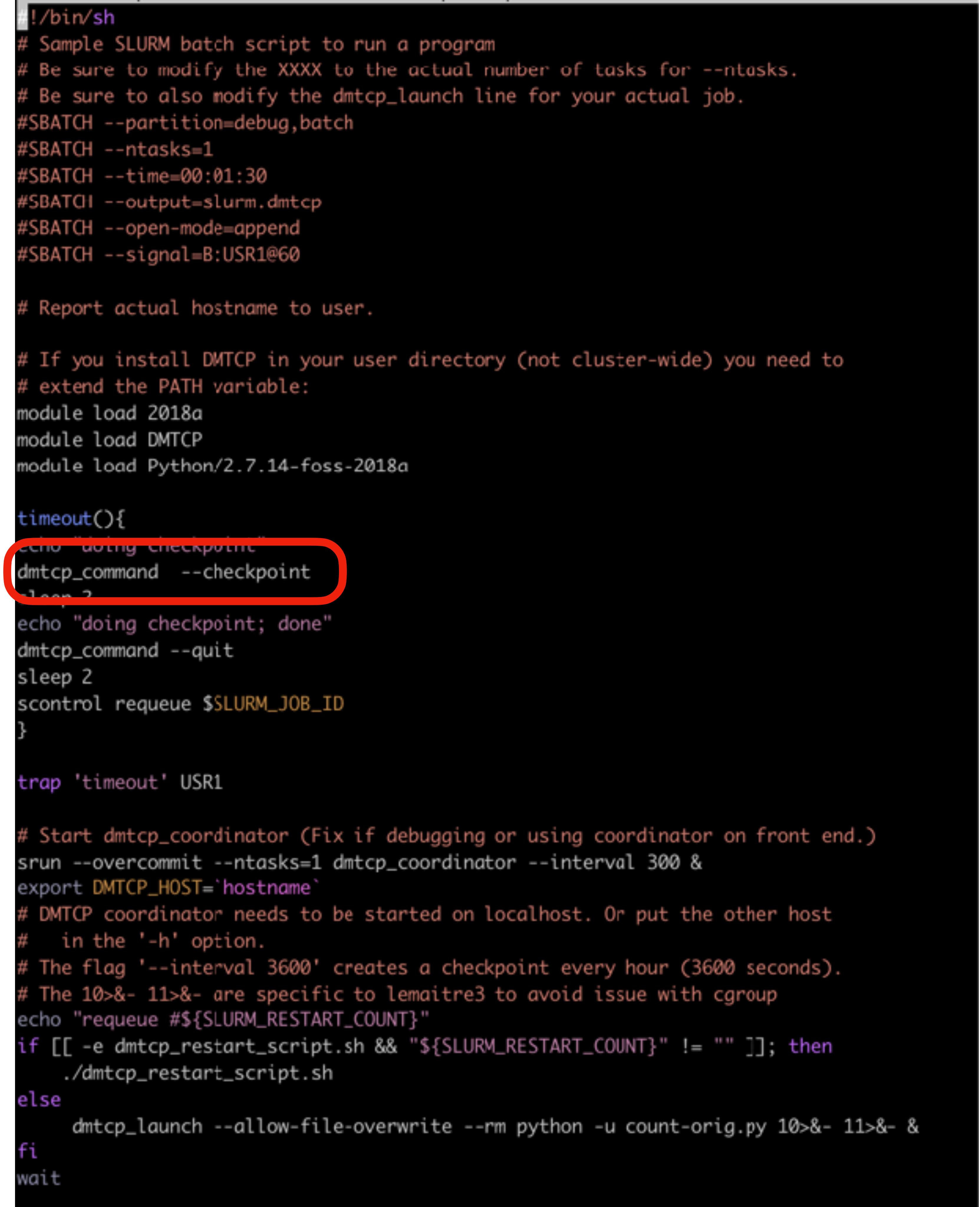


```
#!/bin/sh
# Sample SLURM batch script to run a program
# Be sure to modify the XXXX to the actual number of tasks for --ntasks.
# Be sure to also modify the dmtcp_launch line for your actual job.
#SBATCH --partition=debug,batch
#SBATCH --ntasks=1
#SBATCH --time=00:01:30
#SBATCH --output=slurm.dmtcp
#SBATCH --open-mode=append
#SBATCH --signal=B:USR1@60

# Report actual hostname to user.

# If you install DMTCP in your user directory (not cluster-wide) you need to
# extend the PATH variable:
module load 2018a
module load DMTCP
module load Python/2.7.14-foss-2018a

timeout(){
echo "doing checkpoint"
dmtcp_command  --checkpoint
sleep 2
echo "doing checkpoint; done"
dmtcp_command --quit
sleep 2
scontrol requeue $SLURM_JOB_ID
}

trap 'timeout' USR1

# Start dmtcp_coordinator (Fix if debugging or using coordinator on front end.)
srun --overcommit --ntasks=1 dmtcp_coordinator --interval 300 &
export DMTCP_HOST=`hostname`
# DMTCP coordinator needs to be started on localhost. Or put the other host
#   in the '-h' option.
# The flag '--interval 3600' creates a checkpoint every hour (3600 seconds).
# The 10>&- 11>&- are specific to lemaitre3 to avoid issue with cgroup
echo "requeue #${SLURM_RESTART_COUNT}"
if [[ -e dmtcp_restart_script.sh && "${SLURM_RESTART_COUNT}" != "" ]]; then
    ./dmtcp_restart_script.sh
else
     dmtcp_launch --allow-file-overwrite --rm python -u count-orig.py 10>&- 11>&- &
fi
wait
```

Allow catching

Solution

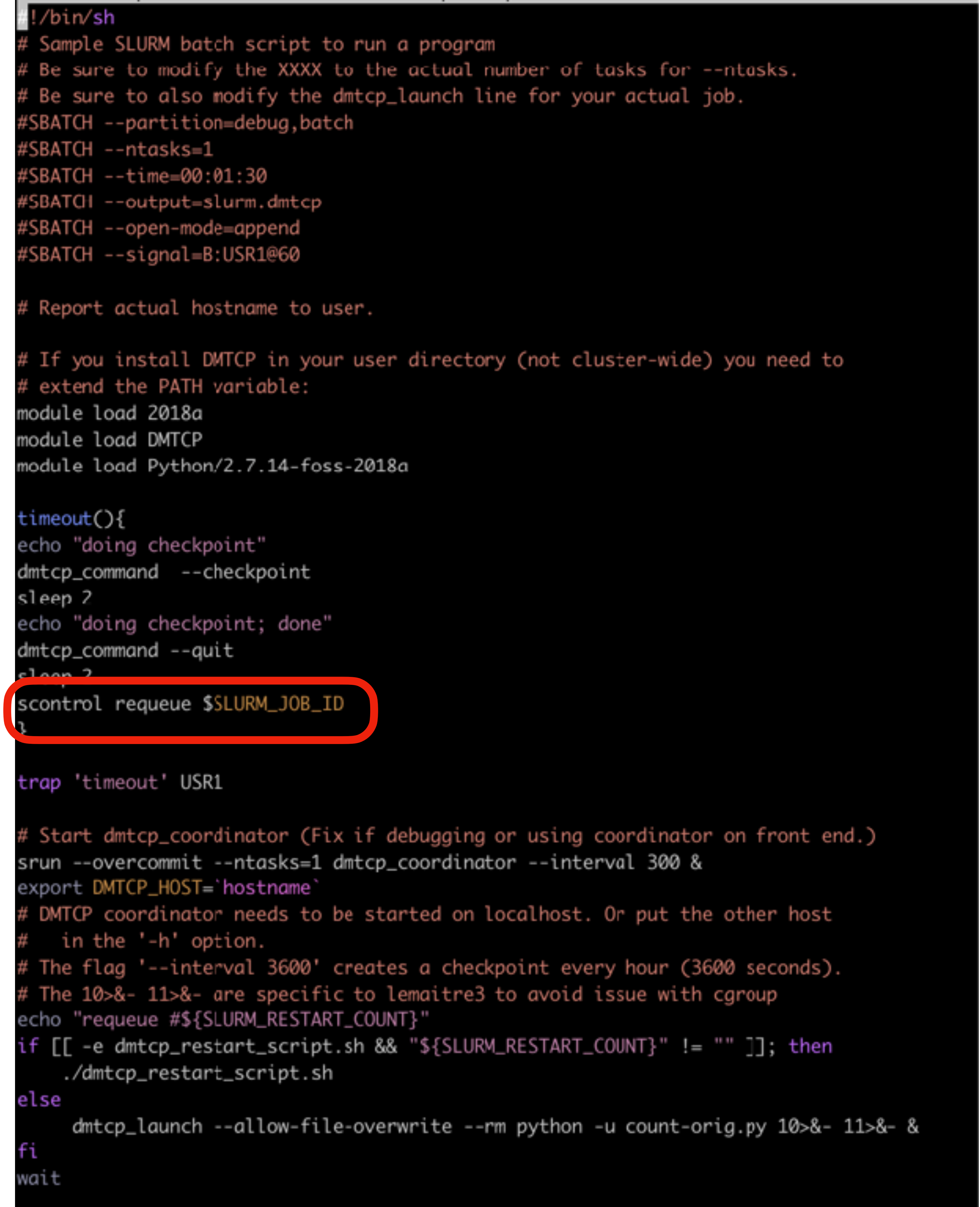
Manual checkpoint



```
#!/bin/sh
# Sample SLURM batch script to run a program
# Be sure to modify the XXXX to the actual number of tasks for --ntasks.
# Be sure to also modify the dmtcp_launch line for your actual job.
#SBATCH --partition=debug,batch
#SBATCH --ntasks=1
#SBATCH --time=00:01:30
#SBATCH --output=slurm.dmtcp
#SBATCH --open-mode=append
#SBATCH --signal=B:USR1@60

# Report actual hostname to user.

# If you install DMTCP in your user directory (not cluster-wide) you need to
# extend the PATH variable:
module load 2018a
module load DMTCP
module load Python/2.7.14-foss-2018a

timeout(){
echo "doing checkpoint"
dmtcp_command  --checkpoint
sleep 2
echo "doing checkpoint; done"
dmtcp_command --quit
sleep 2
scontrol requeue $SLURM_JOB_ID
}

trap 'timeout' USR1

# Start dmtcp_coordinator (Fix if debugging or using coordinator on front end.)
srun --overcommit --ntasks=1 dmtcp_coordinator --interval 300 &
export DMTCP_HOST=`hostname`
# DMTCP coordinator needs to be started on localhost. Or put the other host
#   in the '-h' option.
# The flag '--interval 3600' creates a checkpoint every hour (3600 seconds).
# The 10>&- 11>&- are specific to lemaitre3 to avoid issue with cgroup
echo "requeue #${SLURM_RESTART_COUNT}"
if [[ -e dmtcp_restart_script.sh && "${SLURM_RESTART_COUNT}" != "" ]]; then
    ./dmtcp_restart_script.sh
else
     dmtcp_launch --allow-file-overwrite --rm python -u count-orig.py 10>&- 11>&- &
fi
wait
```

Solution

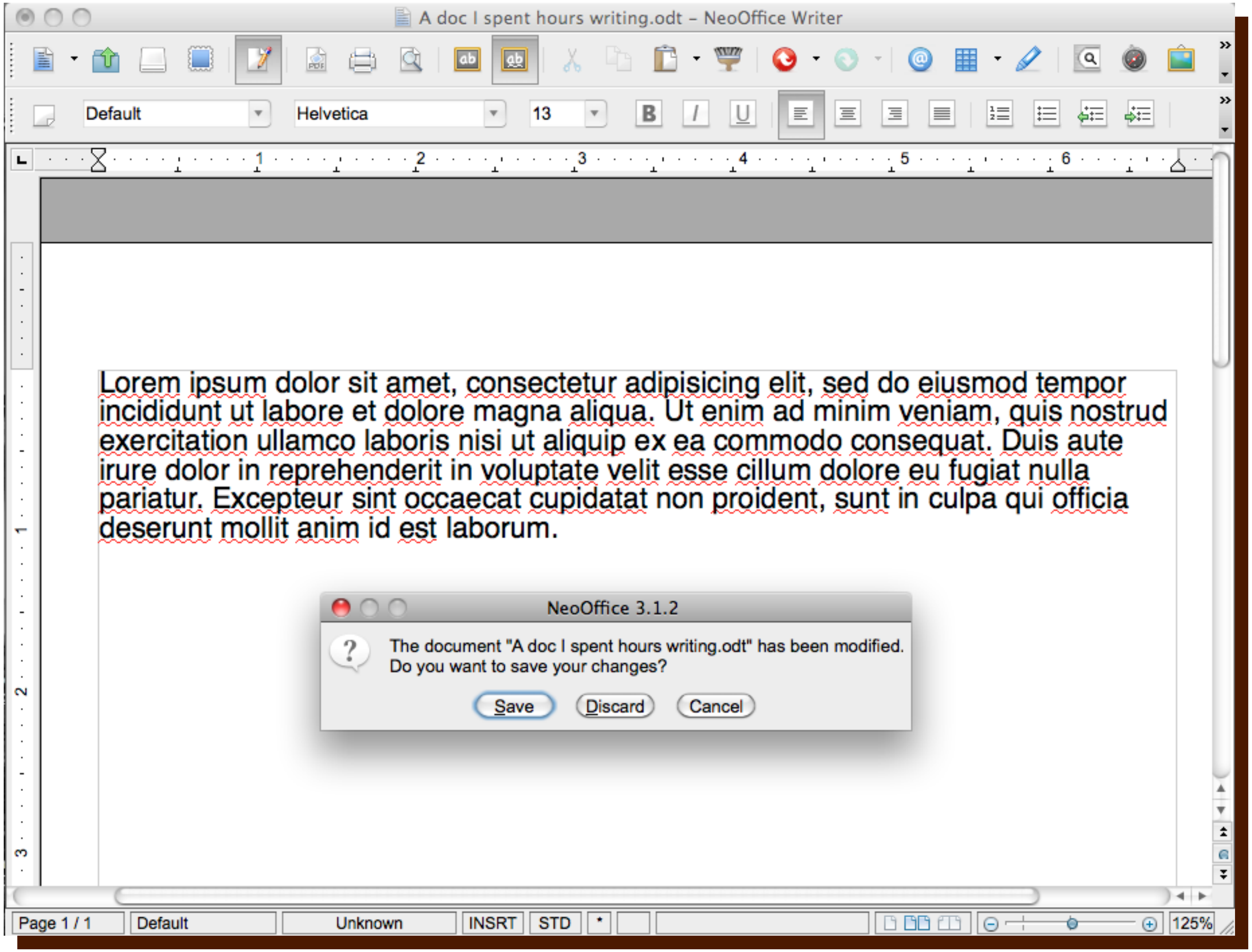
Resubmit



```
#!/bin/sh
# Sample SLURM batch script to run a program
# Be sure to modify the XXXX to the actual number of tasks for --ntasks.
# Be sure to also modify the dmtcp_launch line for your actual job.
#SBATCH --partition=debug,batch
#SBATCH --ntasks=1
#SBATCH --time=00:01:30
#SBATCH --output=slurm.dmtcp
#SBATCH --open-mode=append
#SBATCH --signal=B:USR1@60

# Report actual hostname to user.

# If you install DMTCP in your user directory (not cluster-wide) you need to
# extend the PATH variable:
module load 2018a
module load DMTCP
module load Python/2.7.14-foss-2018a

timeout(){
echo "doing checkpoint"
dmtcp_command  --checkpoint
sleep 2
echo "doing checkpoint; done"
dmtcp_command --quit
sleep 2
scontrol requeue $SLURM_JOB_ID
}

trap 'timeout' USR1

# Start dmtcp_coordinator (Fix if debugging or using coordinator on front end.)
srun --overcommit --ntasks=1 dmtcp_coordinator --interval 300 &
export DMTCP_HOST=`hostname`
# DMTCP coordinator needs to be started on localhost. Or put the other host
#   in the '-h' option.
# The flag '--interval 3600' creates a checkpoint every hour (3600 seconds).
# The 10>&- 11>&- are specific to lemaitre3 to avoid issue with cgroup
echo "requeue #${SLURM_RESTART_COUNT}"
if [[ -e dmtcp_restart_script.sh && "${SLURM_RESTART_COUNT}" != "" ]]; then
    ./dmtcp_restart_script.sh
else
     dmtcp_launch --allow-file-overwrite --rm python -u count-orig.py 10>&- 11>&- &
fi
wait
```

Summary, Wrap-up and Conclusions.

Never click 'Discard' again...



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

NeoOffice 3.1.2

The document "A doc I spent hours writing.odt" has been modified.
Do you want to save your changes?

The submission script(s)

- Either one big one or two small ones
- Checkpoint periodically or --signal
- Requeue automatically
- Open-mode=append