Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'criu.8' command

$ man criu.8

CRIU(8) CRIU Manual CRIU(8)

NAME

criu - checkpoint/restore in userspace

SYNOPSIS

criu command [option ...]

DESCRIPTION

criu is a tool for checkpointing and restoring running applications. It does this by saving their state as a collection of files (see the dump command) and creating equivalent processes from those files (see the restore command). The restore operation can be performed at a later time, on a different system, or both.

OPTIONS

Most of the long flags can be prefixed with no- to negate the option (example: --display-stats and --no-display-stats).

Common options

Common options are applicable to any command.

-v[v...], --verbosity

Increase verbosity up from the default level. In case of short op? tion, multiple v can be used, each increasing verbosity by one.

-vnum, --verbosity=num

Set verbosity level to num. The higher the level, the more output is produced.

The following levels are available:

? -v0 no output;

? -v1 only errors;

? -v2 above plus warnings (this is the default level);

? -v3 above plus information messages and timestamps;

? -v4 above plus lots of debug.

--config file

Pass a specific configuration file to criu.

--no-default-config

Disable parsing of default configuration files.

--pidfile file

Write root task, service or page-server pid into a file.

-o, --log-file file

Write logging messages to a file.

--display-stats

During dump, as well as during restore, criu collects some statis? tics, like the time required to dump or restore the process, or the number of pages dumped or restored. This information is always saved to the stats-dump and stats-restore files, and can be shown using crit(1). The option --display-stats prints out this informa? tion on the console at the end of a dump or restore operation.

-D, --images-dir path

Use path as a base directory where to look for sets of image files.

--stream

dump/restore images using criu-image-streamer. See https://github.com/checkpoint-restore/criu-image-streamer for de? tailed usage.

--prev-images-dir path

Use path as a parent directory where to look for sets of image files. This option makes sense in case of incremental dumps.

-W, --work-dir dir

Use directory dir for putting logs, pidfiles and statistics. If not specified, path from -D option is taken.

--close fd

Close file descriptor fd before performing any actions.

-L, --libdir path

Path to plugins directory.

--enable-fs [fs[,fs...]]

Specify a comma-separated list of filesystem names that should be auto-detected. The value all enables auto-detection for all filesystems.

Note: This option is not safe, use at your own risk. Auto-detecting a filesystem mount assumes that the mountpoint can be restored with mount(src, mountpoint, flags, options). When used, dump is expected to always succeed if a mountpoint is to be auto-detected, however restore may fail (or do something wrong) if the assumption for re? store logic is incorrect. This option is not compatible with --ex? ternal dev.

--action-script script

Add an external action script to be executed at certain stages. The environment variable CRTOOLS_SCRIPT_ACTION is available to the script to find out which action is being executed, and its value can be one of the following:

pre-dump

run prior to beginning a dump

post-dump

run upon dump completion

pre-restore

run prior to beginning a restore

post-restore

run upon restore completion

pre-resume

run when all processes and resources are restored but tasks are stopped waiting for final kick to run. Must not fail.

post-resume

called at the very end, when everything is restored and pro? cesses were resumed

network-lock

run to lock network in a target network namespace

network-unlock

run to unlock network in a target network namespace

setup-namespaces

run once root task has just been created with required name?

spaces. Note it is an early stage of restore, when nothing is

restored yet, except for namespaces themselves

post-setup-namespaces

called after the namespaces are configured

orphan-pts-master

called after master pty is opened and unlocked. This hook can

be used only in the RPC mode, and the notification message con?

tains a file descriptor for the master pty

-V, --version

Print program version and exit.

-h, --help

Print some help and exit.

pre-dump

Performs the pre-dump procedure, during which criu creates a snapshot of memory changes since the previous pre-dump. Note that during this criu also creates the fsnotify cache which speeds up the restore proce? dure. pre-dump requires at least -t option (see dump below). In addi? tion, page-server options may be specified.

--track-mem

Turn on memory changes tracker in the kernel. If the option is not

passed the memory tracker get turned on implicitly.

--pre-dump-mode=mode

There are two mode to operate pre-dump algorithm. The splice mode

is parasite based, whereas read mode is based on process_vm_readv

syscall. The read mode incurs reduced frozen time and reduced mem?

ory pressure as compared to splice mode. Default is splice mode.

dump

Performs a checkpoint procedure.

-t, --tree pid

Checkpoint the whole process tree starting from pid.

-R, --leave-running

Leave tasks in running state after checkpoint, instead of killing.

This option is pretty dangerous and should be used only if you un?

derstand what you are doing.

Note if task is about to run after been checkpointed, it can modify

TCP connections, delete files and do other dangerous actions.

Therefore, criu can not guarantee that the next restore action will

succeed. Most likely if this option is used, at least the file sys?

tem snapshot must be made with the help of post-dump action script.

In other words, do not use it unless really needed.

-s, --leave-stopped

Leave tasks in stopped state after checkpoint, instead of killing.

--external type[id]:value

Dump an instance of an external resource. The generic syntax is

type of resource, followed by resource id (enclosed in literal

square brackets), and optional value (prepended by a literal

colon). The following resource types are currently supported: mnt,

dev, file, tty, unix. Syntax depends on type. Note to restore ex?

ternal resources, either --external or --inherit-fd is used, de?

pending on resource type.

--external mnt[mountpoint]:name

Dump an external bind mount referenced by mountpoint, saving it to

image under the identifier name.

--external mnt[]:flags

Dump all external bind mounts, autodetecting those. Optional flags

can contain m to also dump external master mounts, s to also dump

external shared mounts (default behavior is to abort dumping if

such mounts are found). If flags are not provided, colon is op?

tional.

--external dev[major/minor]:name

Allow to dump a mount namespace having a real block device mounted.

A block device is identified by its major and minor numbers, and

criu saves its information to image under the identifier name.

--external file[mnt_id:inode]

Dump an external file, i.e. an opened file that is can not be re?

solved from the current mount namespace, which can not be dumped

without using this option. The file is identified by mnt_id (a

field obtained from /proc/pid/fdinfo/N) and inode (as returned by

stat(2)).

--external tty[rdev:dev]

Dump an external TTY, identified by st_rdev and st_dev fields re?

turned by stat(2).

--external unix[id]

Tell criu that one end of a pair of UNIX sockets (created by sock?

etpair(2)) with the given id is OK to be disconnected.

--external net[inode]:name

Mark a network namespace as external and do not include it in the

checkpoint. The label name can be used with --inherit-fd during re?

store to specify a file descriptor to a preconfigured network name?

space.

--external pid[inode]:name

Mark a PID namespace as external. This can be later used to restore

a process into an existing PID namespace. The label name can be

used to assign another PID namespace during restore with the help

of --inherit-fd.

--freeze-cgroup

Use cgroup freezer to collect processes.

--manage-cgroups

Collect cgroups into the image thus they gonna be restored then.

Without this option, criu will not save cgroups configuration asso?

ciated with a task.

--cgroup-props spec

Specify controllers and their properties to be saved into the image

file. criu predefines specifications for common controllers, but since the kernel can add new controllers and modify their proper? ties, there should be a way to specify ones matched the kernel. spec argument describes the controller and properties specification in a simplified YAML form:

```
"c1":
 - "strategy": "merge"
 - "properties": ["a", "b"]
"c2":
 - "strategy": "replace"
 - "properties": ["c", "d"]
```

where c1 and c2 are controllers names, and a, b, c, d are their properties.

Note the format: double quotes, spaces and new lines are required.

The strategy specifies what to do if a controller specified already exists as a built-in one: criu can either merge or replace such.

For example, the command line for the above example should look like this:

```
--cgroup-props "\"c1\":\n - \"strategy\": \"merge\"\n - \"properties\": [\"a\", \"b\"]\n \"c2\":\n - \"strategy\": \"replace\"\n - \"properties\": [\"c\", \"d\"]"
```

--cgroup-props-file file

Same as --cgroup-props, except the specification is read from the file.

--cgroup-dump-controller name

Dump a controller with name only, skipping anything else that was discovered automatically (usually via /proc). This option is useful when one needs criu to skip some controllers.

--cgroup-yard path

Instead of trying to mount cgroups in CRIU, provide a path to a di? rectory with already created cgroup yard. Useful if you don?t want to grant CAP_SYS_ADMIN to CRIU. For every cgroup mount there should be exactly one directory. If there is only one controller in this mount, the dir?s name should be just the name of the controller. If

there are multiple controllers comounted, the directory name should have them be separated by a comma.

For example, if /proc/cgroups looks like this:

```
#subsys_name hierarchy num_cgroups enabled
cpu          1         1           1
devices      2         2           1
freezer      2         2           1
```

then you can create the cgroup yard by the following commands:

```
mkdir private_yard
cd private_yard
mkdir cpu
mount -t cgroup -o cpu none cpu
mkdir devices,freezer
mount -t cgroup -o devices,freezer none devices,freezer
```

--tcp-established

Checkpoint established TCP connections.

--tcp-close

Don?t dump the state of, or block, established tcp connections (in? cluding the connection is once established but now closed). This is useful when tcp connections are not going to be restored.

--skip-in-flight

This option skips in-flight TCP connections. If any TCP connections that are not yet completely established are found, criu ignores these connections, rather than errors out. The TCP stack on the client side is expected to handle the re-connect gracefully.

--evasive-devices

Use any path to a device file if the original one is inaccessible.

--page-server

Send pages to a page server (see the page-server command).

--force-irmap

Force resolving names for inotify and fsnotify watches.

--auto-dedup

Deduplicate "old" data in pages images of previous dump. This op?

tion implies incremental dump mode (see the pre-dump command).

-l, --file-locks

Dump file locks. It is necessary to make sure that all file lock users are taken into dump, so it is only safe to use this for en? closed containers where locks are not held by any processes outside of dumped process tree.

--link-remap

Allows to link unlinked files back, if possible (modifies filesys? tem during restore).

--timeout number

Set a time limit in seconds for collecting tasks during the dump operation. The timeout is 10 seconds by default.

--ghost-limit size

Set the maximum size of deleted file to be carried inside image. By default, up to 1M file is allowed. Using this option allows to not put big deleted files inside images. Argument size may be postfixed with a K, M or G, which stands for kilo-, mega, and gigabytes, ac? cordingly.

-j, --shell-job

Allow one to dump shell jobs. This implies the restored task will inherit session and process group ID from the criu itself. This op? tion also allows to migrate a single external tty connection, to migrate applications like top. If used with dump command, it must be specified with restore as well.

--cpu-cap [cap[,cap...]]

Specify CPU capabilities to write to an image file. The argument is a comma-separated list of:

? none to ignore capabilities at all; the image will not be pro? duced on dump, neither any check performed on restore;

? fpu to check if FPU module is compatible;

? ins to check if CPU supports all instructions required;

? cpu to check if CPU capabilities are exactly matching;

? all for all above set.

By default the option is set to fpu and ins.

--cgroup-root [controller:]/newroot

Change the root for the controller that will be dumped. By default, criu simply dumps everything below where any of the tasks live. However, if a container moves all of its tasks into a cgroup direc? tory below the container engine?s default directory for tasks, per? missions will not be preserved on the upper directories with no tasks in them, which may cause problems.

--lazy-pages

Perform the dump procedure without writing memory pages into the image files and prepare to service page requests over the network. When dump runs in this mode it presumes that lazy-pages daemon will connect to it and fetch memory pages to lazily inject them into the restored process address space. This option is intended for post-copy (lazy) migration and should be used in conjunction with restore with appropriate options.

--file-validation [mode]

Set the method to be used to validate open files. Validation is done to ensure that the version of the file being restored is the same version when it was dumped.

The mode may be one of the following:

filesize

To explicitly use only the file size check all the time. This is the fastest and least intensive check.

buildid

To validate ELF files with their build-ID. If the build-ID can? not be obtained, chksm-first method will be used. This is the default if mode is unspecified.

--network-lock [mode]

Set the method to be used for network locking/unlocking. Locking is done to ensure that tcp packets are dropped between dump and re? store. This is done to avoid the kernel sending RST when a packet arrives destined for the dumped process.

The mode may be one of the following:

iptables

Use iptables rules to drop the packets. This is the default if mode is not specified.

nftables

Use nftables rules to drop the packets.

restore

Restores previously checkpointed processes.

--inherit-fd fd[N]:resource

Inherit a file descriptor. This option lets criu use an already opened file descriptor N for restoring a file identified by re?source. This option can be used to restore an external resource dumped with the help of --external file, tty, pid and unix options.

The resource argument can be one of the following:

? tty[rdev:dev]

? pipe[inode]

? socket[inode*]*

? file[mnt_id:inode]

? path/to/file

Note that square brackets used in this option arguments are liter?als and usually need to be escaped from shell.

-d, --restore-detached

Detach criu itself once restore is complete.

-s, --leave-stopped

Leave tasks in stopped state after restore (rather than resuming their execution).

-S, --restore-sibling

Restore root task as a sibling (makes sense only with --restore-de?tached).

--log-pid

Write separate logging files per each pid.

-r, --root path

Change the root filesystem to path (when run in a mount namespace).

This option is required to restore a mount namespace. The directory path must be a mount point and its parent must not be overmounted.

--external type[id]:value

Restore an instance of an external resource. The generic syntax is type of resource, followed by resource id (enclosed in literal square brackets), and optional value (prepended by a literal colon). The following resource types are currently supported: mnt, dev, veth, macvlan. Syntax depends on type. Note to restore exter? nal resources dealing with opened file descriptors (such as dumped with the help of --external file, tty, and unix options), option --inherit-fd should be used.

--external mnt[name]:mountpoint

Restore an external bind mount referenced in the image by name, bind-mounting it from the host mountpoint to a proper mount point.

--external mnt[]

Restore all external bind mounts (dumped with the help of --exter? nal mnt[] auto-detection).

--external dev[name]:/dev/path

Restore an external mount device, identified in the image by name, using the existing block device /dev/path.

--external veth[inner_dev]:outer_dev@bridge

Set the outer VETH device name (corresponding to inner_dev being restored) to outer_dev. If optional @bridge is specified, outer_dev is added to that bridge. If the option is not used, outer_dev will be autogenerated by the kernel.

--external macvlan[inner_dev]:outer_dev

When restoring an image that have a MacVLAN device in it, this op? tion must be used to specify to which outer_dev (an existing net? work device in CRIU namespace) the restored inner_dev should be bound to.

-J, --join-ns NS:{PID|NS_FILE}[,EXTRA_OPTS]

Restore process tree inside an existing namespace. The namespace can be specified in PID or NS_FILE path format (example: --join-ns

net:12345 or --join-ns net:/foo/bar). Currently supported values for NS are: ipc, net, time, user, and uts. This option doesn?t support joining a PID namespace, however, this is possible using --external and --inheritfd. EXTRA_OPTS is optional and can be used to specify UID and GID for user namespace (e.g., --join-ns user:PID,UID,GID).

--manage-cgroups [mode]

Restore cgroups configuration associated with a task from the image. Controllers are always restored in an optimistic way ? if already present in system, criu reuses it, otherwise it will be created.

The mode may be one of the following:

none

Do not restore cgroup properties but require cgroup to pre-exist at the moment of restore procedure.

props

Restore cgroup properties and require cgroup to pre-exist.

soft

Restore cgroup properties if only cgroup has been created by criu, otherwise do not restore properties. This is the default if mode is unspecified.

full

Always restore all cgroups and their properties.

strict

Restore all cgroups and their properties from the scratch, requiring them to not present in the system.

ignore

Don?t deal with cgroups and pretend that they don?t exist.

--cgroup-yard path

Instead of trying to mount cgroups in CRIU, provide a path to a directory with already created cgroup yard. For more information look in the dump section.

--cgroup-root [controller:]/newroot

Change the root cgroup the controller will be installed into. No controller means that root is the default for all controllers not specified.

--tcp-established

Restore previously dumped established TCP connections. This implies that the network has been locked between dump and restore phases so other side of a connection simply notice a kind of lag.

--tcp-close

Restore connected TCP sockets in closed state.

--veth-pair IN=OUT

Correspondence between outside and inside names of veth devices.

-l, --file-locks

Restore file locks from the image.

--lsm-profile type:name

Specify an LSM profile to be used during restore. The type can be either apparmor or selinux.

--lsm-mount-context context

Specify a new mount context to be used during restore.

This option will only replace existing mount context information with the one specified with this option. Mounts without the con? text= option will not be changed.

If a mountpoint has been checkpointed with an option like

context="system_u:object_r:container_file_t:s0:c82,c137"

it is possible to change this option using

--lsm-mount-context "system_u:object_r:container_file_t:s0:c204,c495"

which will result that the mountpoint will be restored with the new context=.

This option is useful if using selinux and if the selinux labels need to be changed on restore like if a container is restored into an existing Pod.

--auto-dedup

As soon as a page is restored it get punched out from image.

-j, --shell-job

Restore shell jobs, in other words inherit session and process group ID from the criu itself.

--cpu-cap [cap[,cap...]]

Specify CPU capabilities to be present on the CPU the process is restoring. To inverse a capability, prefix it with ^. This option implies that --cpu-cap has been passed on dump as well, except fpu option case. The cap argument can be the following (or a set of comma-separated values):

all

Require all capabilities. This is default mode if --cpu-cap is passed without arguments. Most safe mode.

cpu

Require the CPU to have all capabilities in image to match run? time CPU.

fpu

Require the CPU to have compatible FPU. For example the process might be dumped with xsave capability but attempted to restore without it present on target CPU. In such case we refuse to proceed. This is default mode if --cpu-cap is not present in command line. Note this argument might be passed even if on the dump no --cpu-cap have been specified because FPU frames are always encoded into images.

ins

Require CPU compatibility on instructions level.

none

Ignore capabilities. Most dangerous mode. The behaviour is im? plementation dependent. Try to not use it until really re? quired.

For example, this option can be used in case --cpu-cap=cpu was used during dump, and images are migrated to a less capable CPU and are to be restored. By default, criu shows an error that CPU capabilities are not adequate, but this can be suppressed by using --cpu-cap=none.

--weak-sysctls

Silently skip restoring sysctls that are not available. This allows to restore on an older kernel, or a kernel configured without some options.

--lazy-pages

Restore the processes without filling out the entire memory con? tents. When this option is used, restore sets up the infrastructure required to fill memory pages either on demand when the process ac? cesses them or in the background without stopping the restored process. This option requires running lazy-pages daemon.

--file-validation [mode]

Set the method to be used to validate open files. Validation is done to ensure that the version of the file being restored is the same version when it was dumped.

The mode may be one of the following:

filesize

To explicitly use only the file size check all the time. This is the fastest and least intensive check.

buildid

To validate ELF files with their build-ID. If the build-ID can? not be obtained, chksm-first method will be used. This is the default if mode is unspecified.

check

Checks whether the kernel supports the features needed by criu to dump and restore a process tree.

There are three categories of kernel support, as described below. criu check always checks Category 1 features unless --feature is specified which only checks a specified feature.

Category 1

Absolutely required. These are features like support for /proc/PID/map_files, NETLINK_SOCK_DIAG socket monitoring, /proc/sys/kernel/ns_last_pid etc.

Category 2

Required only for specific cases. These are features like AIO remap, /dev/net/tun and others that are only required if a process being dumped or restored is using those.

Category 3

Experimental. These are features like task-diag that are used for experimental purposes (mostly during development).

If there are no errors or warnings, criu prints "Looks good." and its exit code is 0.

A missing Category 1 feature causes criu to print "Does not look good." and its exit code is non-zero.

Missing Category 2 and 3 features cause criu to print "Looks good but ..." and its exit code is be non-zero.

Without any options, criu check checks Category 1 features. This behav? ior can be changed by using the following options:

--extra

Check kernel support for Category 2 features.

--experimental

Check kernel support for Category 3 features.

--all

Check kernel support for Category 1, 2, and 3 features.

--feature name

Check a specific feature. If name is list, a list of valid kernel feature names that can be checked will be printed.

page-server

Launches criu in page server mode.

--daemon

Runs page server as a daemon (background process).

--status-fd

Write \0 to the FD and close it once page-server is ready to handle requests. The status-fd allows to not daemonize a process and get its exit code at the end. It isn?t supposed to use --daemon and --status-fd together.

--address address

Page server IP address or hostname.

--port number

Page server port number.

--ps-socket fd

Use provided file descriptor as socket for incoming connection. In this case --address and --port are ignored. Useful for intercepting page-server traffic e.g. to add encryption or authentication.

--lazy-pages

Serve local memory dump to a remote lazy-pages daemon. In this mode the page-server reads local memory dump and allows the remote lazy-pages daemon to request memory pages in random order.

--tls-cacert file

Specifies the path to a trusted Certificate Authority (CA) certifi? cate file to be used for verification of a client or server cer? tificate. The file must be in PEM format. When this option is used only the specified CA is used for verification. Otherwise, the sys? tem?s trusted CAs and, if present, /etc/pki/CA/cacert.pem will be used.

--tls-cacrl file

Specifies a path to a Certificate Revocation List (CRL) file which contains a list of revoked certificates that should no longer be trusted. The file must be in PEM format. When this option is not specified, the file, if present, /etc/pki/CA/cacrl.pem will be used.

--tls-cert file

Specifies a path to a file that contains a X.509 certificate to present to the remote entity. The file must be in PEM format. When this option is not specified, the default location (/etc/pki/criu/cert.pem) will be used.

--tls-key file

Specifies a path to a file that contains TLS private key. The file must be in PEM format. When this option is not the default location (/etc/pki/criu/private/key.pem) will be used.

--tls

Use TLS to secure remote connections.

lazy-pages

Launches criu in lazy-pages daemon mode.

The lazy-pages daemon is responsible for managing user-level demand paging for the restored processes. It gets information required to fill the process memory pages from the restore and from the checkpoint di? rectory. When a restored process access certain memory page for the first time, the lazy-pages daemon injects its contents into the process address space. The memory pages that are not yet requested by the re? stored processes are injected in the background.

exec

Executes a system call inside a destination task's context. This func? tionality is deprecated; please use Compel instead.

service

Launches criu in RPC daemon mode, where criu is listening for RPC com? mands over socket to perform. This is convenient for a case where dae? mon itself is running in a privileged (superuser) mode but clients are not.

dedup

Starts pagemap data deduplication procedure, where criu scans over all pagemap files and tries to minimize the number of pagemap entries by obtaining the references from a parent pagemap image.

cpuinfo dump

Fetches current CPU features and write them into an image file.

cpuinfo check

Fetches current CPU features (i.e. CPU the criu is running on) and test if they are compatible with the ones present in an image file.

CONFIGURATION FILES

Criu supports usage of configuration files to avoid the need of writing every option on command line, which is useful especially with repeated usage of same options. A specific configuration file can be passed with the "--config file" option. If no file is passed, the default configu?

ration files /etc/criu/default.conf and $HOME/.criu/default.conf are parsed (if present on the system). If the environment variable CRIU_CONFIG_FILE is set, it will also be parsed.

The options passed to CRIU via CLI, RPC or configuration file are eval? uated in the following order:

? apply_config(/etc/criu/default.conf)

? apply_config($HOME/.criu/default.conf)

? apply_config(CRIU_CONFIG_FILE)

? apply_config(--config file)

? apply_config(CLI) or apply_config(RPC)

? apply_config(RPC configuration file) (only for RPC mode)

Default configuration file parsing can be deactivated with "--no-de? fault-config" if needed. Parsed configuration files are merged with command line options, which allows overriding boolean options.

Configuration file syntax

Comments are supported using '#' sign. The rest of the line is ignored. Options are the same as command line options without the '--' prefix, use one option per line (with corresponding argument if applicable, di? vided by whitespaces). If needed, the argument can be provided in dou? ble quotes (this should be needed only if the argument contains white? spaces). In case this type of argument contains a literal double quote as well, it can be escaped using the '\' sign. Usage of commands is disallowed and all other escape sequences are interpreted literally.

Example of configuration file to illustrate syntax:

```
$ cat ~/.criu/default.conf
tcp-established
work-dir "/home/USERNAME/criu/my \"work\" directory"
#this is a comment
no-restore-sibling   # this is another comment
```

Configuration files in RPC mode

Not only does criu evaluate configuration files in CLI mode, it also evaluates configuration files in RPC mode. Just as in CLI mode the con? figuration file values are evaluated first. This means that any option

set via RPC will overwrite the configuration file setting. The user can thus change criu's default behavior but it is not possible to change settings which are explicitly set by the RPC client.

The RPC client can, however, specify an additional configuration file which will be evaluated after the RPC options (see above for option evaluation order). The RPC client can specify this additional configu? ration file via "req.opts.config_file = /path/to/file". The values from this configuration file will overwrite all other configuration file settings or RPC options. This can lead to undesired behavior of criu and should only be used carefully.

EXAMPLES

To checkpoint a program with pid of 1234 and write all image files into directory checkpoint:

```
criu dump -D checkpoint -t 1234
```

To restore this program detaching criu itself:

```
criu restore -d -D checkpoint
```

AUTHOR

The CRIU team.

COPYRIGHT