NAME

env_parallel - export environment to GNU parallel

SYNOPSIS

env_parallel [options for GNU Parallel]

DESCRIPTION

env_parallel is a shell function that exports the current environment to GNU **parallel**.

If the shell function is not loaded, a dummy script will be run instead that explains how to install the function.

env_parallel is beta quality and not production ready, but please use it for everyday use and report bugs.

env_parallel is 100 ms slower at startup than pure GNU **parallel**, and takes up to 30% longer to start a job (typically 15 ms).

Due to the problem with environment space (see below) the recommended usage is either:

```
# Do --record-env into $PARALLEL_IGNORED_NAMES
env_parallel --session

# Define whatever you want to use
alias myalias=echo
myvar=it
myfunc() { myalias $1 $myvar works.; }

# env_parallel will not export names in $PARALLEL_IGNORED_NAMES
env_parallel -S localhost myfunc ::: Yay,
```

Or:

```
# Record the "clean" environment (this only needs to be run once)
env_parallel --record-env

# Optionally edit ~/.parallel/ignored_vars (only needed once)

# Define whatever you want to use
alias myalias=echo
myvar=it
myfunc() { myalias $1 $myvar works.; }

# Use --env _ to only transfer the names not in the "empty" environment
env_parallel --env _ -S localhost myfunc ::: Yay,
```

In **csh --session** is not supported:

```
# Record the "clean" environment (this only needs to be run once)
env_parallel --record-env

# Optionally edit ~/.parallel/ignored_vars (only needed once)

# Define whatever you want to use
alias myalias 'echo \!* $myvar works.'
set myvar=it
```

```
# Use --env _ to only transfer the names not in the "empty" environment
env_parallel --env _ -S localhost myalias ::: Yay,
```

Environment space

By default **env_parallel** will export all environment variables, arrays, aliases, functions and shell options (see details for the individual shells below).

But this only works if the size of the current environment is smaller than the maximal length of a command and smaller than half of the max if running remotely. E.g. The max size of Bash's command is 128 KB, so **env_parallel** will fail if '**set | wc -c**' is bigger than 128 KB. Technically the limit is in execve(1) which IPC::open3 uses.

Bash completion functions are well-known for taking up well over 128 KB of environment space and the primary reason for causing **env_parallel** to fail.

Instead you can use **--env** to specify which variables, arrays, aliases and functions to export as this will only export those with the given name. Or follow the recommended usage in shown in DESCRIPTION.

OPTIONS

Same as GNU **parallel**.

SUPPORTED SHELLS

Ash

Installation

Put this in $HOME/.profile:

```
. `which env_parallel.ash`
```

E.g. by doing:

```
echo '. `which env_parallel.ash`' >> $HOME/.profile
```

Supported use

--env is supported to export only the variable, or alias with the given name. Multiple **--env**s can be given.

aliases

```
alias myecho='echo aliases'
env_parallel myecho ::: work
env_parallel -S server myecho ::: work
env_parallel --env myecho myecho ::: work
env_parallel --env myecho -S server myecho ::: work

alias multiline='echo multiline
  echo aliases'
env_parallel multiline ::: work
env_parallel -S server multiline ::: work
env_parallel --env multiline multiline ::: work
env_parallel --env multiline -S server multiline ::: work
```

functions

```
ash cannot list defined functions - thus is not supported.
```

variables

```
myvar=variables
env_parallel echo '$myvar' ::: work
env_parallel -S server echo '$myvar' ::: work
env_parallel --env myvar echo '$myvar' ::: work
env_parallel --env myvar -S server echo '$myvar' ::: work
```

arrays

Arrays are not supported by Ash.

Bash

Installation

Put this in $HOME/.bashrc:

```
. `which env_parallel.bash`
```

E.g. by doing:

```
echo '. `which env_parallel.bash`' >> $HOME/.bashrc
```

Supported use

--env is supported to export only the variable, alias, function, or array with the given name. Multiple **--env**s can be given.

aliases

```
alias myecho='echo aliases'
env_parallel myecho ::: work
env_parallel -S server myecho ::: work
env_parallel --env myecho myecho ::: work
env_parallel --env myecho -S server myecho ::: work

alias multiline='echo multiline
  echo aliases'
env_parallel 'multiline {};
  echo but only when followed by a newline' ::: work
env_parallel -S server 'multiline {};
  echo but only when followed by a newline' ::: work
env_parallel --env multiline 'multiline {};
  echo but only when followed by a newline' ::: work
env_parallel --env multiline -S server 'multiline {};
  echo but only when followed by a newline' ::: work
```

functions

```
myfunc() { echo functions $*; }
env_parallel myfunc ::: work
env_parallel -S server myfunc ::: work
env_parallel --env myfunc myfunc ::: work
env_parallel --env myfunc -S server myfunc ::: work
```

variables

```
myvar=variables
env_parallel echo '$myvar' ::: work
env_parallel -S server echo '$myvar' ::: work
env_parallel --env myvar echo '$myvar' ::: work
env_parallel --env myvar -S server echo '$myvar' ::: work
```

arrays

```
myarray=(arrays work, too)
env_parallel -k echo '${myarray[{}]}' ::: 0 1 2
env_parallel -k -S server echo '${myarray[{}]}' ::: 0 1 2
env_parallel -k --env myarray echo '${myarray[{}]}' ::: 0 1 2
env_parallel -k --env myarray -S server \
  echo '${myarray[{}]}' ::: 0 1 2
```

BUGS

Due to a bug in Bash, aliases containing newlines must be followed by a newline in the command.

csh

env_parallel for **csh** breaks **$PARALLEL**, so do not use **$PARALLEL**.

Installation

Put this in $HOME/.cshrc:

```
source `which env_parallel.csh`
```

E.g. by doing:

```
echo 'source `which env_parallel.csh`' >> $HOME/.cshrc
```

Supported use

--env is supported to export only the variable, alias, or array with the given name. Multiple **--env**s can be given.

aliases

```
alias myecho 'echo aliases'
env_parallel myecho ::: work
env_parallel -S server myecho ::: work
env_parallel --env myecho myecho ::: work
env_parallel --env myecho -S server myecho ::: work
```

functions

Not supported by **csh**.

variables

```
set myvar=variables
env_parallel echo '$myvar' ::: work
env_parallel -S server echo '$myvar' ::: work
env_parallel --env myvar echo '$myvar' ::: work
env_parallel --env myvar -S server echo '$myvar' ::: work
```

arrays with no special chars

```
set myarray=(arrays work, too)
env_parallel -k echo \$'{myarray[{}]}' ::: 1 2 3
env_parallel -k -S server echo \$'{myarray[{}]}' ::: 1 2 3
env_parallel -k --env myarray echo \$'{myarray[{}]}' ::: 1 2 3
env_parallel -k --env myarray -S server \
  echo \$'{myarray[{}]}' ::: 1 2 3
```

Dash

Installation

Put this in $HOME/.profile:

```
. `which env_parallel.dash`
```

E.g. by doing:

```
echo '. `which env_parallel.dash`' >> $HOME/.profile
```

Supported use

--env is supported to export only the variable, or alias with the given name. Multiple **--env**s can be given.

aliases

```
alias myecho='echo aliases'
env_parallel myecho ::: work
env_parallel -S server myecho ::: work
env_parallel --env myecho myecho ::: work
env_parallel --env myecho -S server myecho ::: work

alias multiline='echo multiline
  echo aliases'
env_parallel multiline ::: work
env_parallel -S server multiline ::: work
env_parallel --env multiline multiline ::: work
env_parallel --env multiline -S server multiline ::: work
```

functions

```
dash cannot list defined functions - thus is not supported.
```

variables

```
myvar=variables
env_parallel echo '$myvar' ::: work
env_parallel -S server echo '$myvar' ::: work
env_parallel --env myvar echo '$myvar' ::: work
env_parallel --env myvar -S server echo '$myvar' ::: work
```

arrays

```
dash does not support arrays.
```

fish

Installation

Put this in $HOME/.config/fish/config.fish:

```
source (which env_parallel.fish)
```

E.g. by doing:

```
echo 'source (which env_parallel.fish)' \
  >> $HOME/.config/fish/config.fish
```

Supported use

--env is supported to export only the variable, alias, function, or array with the given name. Multiple **--env**s can be given.

aliases

```
alias myecho 'echo aliases'
env_parallel myecho ::: work
env_parallel -S server myecho ::: work
env_parallel --env myecho myecho ::: work
env_parallel --env myecho -S server myecho ::: work
```

functions

```
function myfunc
  echo functions $argv
end
env_parallel myfunc ::: work
env_parallel -S server myfunc ::: work
env_parallel --env myfunc myfunc ::: work
env_parallel --env myfunc -S server myfunc ::: work
```

variables

```
set myvar variables
env_parallel echo '$myvar' ::: work
env_parallel -S server echo '$myvar' ::: work
env_parallel --env myvar echo '$myvar' ::: work
env_parallel --env myvar -S server echo '$myvar' ::: work
```

arrays

```
set myarray arrays work, too
env_parallel -k echo '$myarray[{}]' ::: 1 2 3
env_parallel -k -S server echo '$myarray[{}]' ::: 1 2 3
env_parallel -k --env myarray echo '$myarray[{}]' ::: 1 2 3
env_parallel -k --env myarray -S server \
  echo '$myarray[{}]' ::: 1 2 3
```

ksh

Installation

Put this in $HOME/.kshrc:

```
source `which env_parallel.ksh`
```

E.g. by doing:

```
echo 'source `which env_parallel.ksh`' >> $HOME/.kshrc
```

Supported use

--env is supported to export only the variable, alias, function, or array with the given name. Multiple **--env**s can be given.

aliases

```
alias myecho='echo aliases'
env_parallel myecho ::: work
env_parallel -S server myecho ::: work
```

```
env_parallel --env myecho myecho ::: work
env_parallel --env myecho -S server myecho ::: work

alias multiline='echo multiline
  echo aliases'
env_parallel multiline ::: work
env_parallel -S server multiline ::: work
env_parallel --env multiline multiline ::: work
env_parallel --env multiline -S server multiline ::: work
```

functions

```
myfunc() { echo functions $*; }
env_parallel myfunc ::: work
env_parallel -S server myfunc ::: work
env_parallel --env myfunc myfunc ::: work
env_parallel --env myfunc -S server myfunc ::: work
```

variables

```
myvar=variables
env_parallel echo '$myvar' ::: work
env_parallel -S server echo '$myvar' ::: work
env_parallel --env myvar echo '$myvar' ::: work
env_parallel --env myvar -S server echo '$myvar' ::: work
```

arrays

```
myarray=(arrays work, too)
env_parallel -k echo '${myarray[{}]}' ::: 0 1 2
env_parallel -k -S server echo '${myarray[{}]}' ::: 0 1 2
env_parallel -k --env myarray echo '${myarray[{}]}' ::: 0 1 2
env_parallel -k --env myarray -S server \
  echo '${myarray[{}]}' ::: 0 1 2
```

mksh (beta testing)

Installation

Put this in $HOME/.mkshrc:

```
source `which env_parallel.mksh`
```

E.g. by doing:

```
echo 'source `which env_parallel.mksh`' >> $HOME/.mkshrc
```

Supported use

--env is supported to export only the variable, alias, function, or array with the given name. Multiple **--env**s can be given.

aliases

```
alias myecho='echo aliases'
env_parallel myecho ::: work
env_parallel -S server myecho ::: work
env_parallel --env myecho myecho ::: work
env_parallel --env myecho -S server myecho ::: work

alias multiline='echo multiline
```

```
  echo aliases'
env_parallel multiline ::: work
env_parallel -S server multiline ::: work
env_parallel --env multiline multiline ::: work
env_parallel --env multiline -S server multiline ::: work
```

functions

```
myfunc() { echo functions $*; }
env_parallel myfunc ::: work
env_parallel -S server myfunc ::: work
env_parallel --env myfunc myfunc ::: work
env_parallel --env myfunc -S server myfunc ::: work
```

variables

```
myvar=variables
env_parallel echo '$myvar' ::: work
env_parallel -S server echo '$myvar' ::: work
env_parallel --env myvar echo '$myvar' ::: work
env_parallel --env myvar -S server echo '$myvar' ::: work
```

arrays

```
myarray=(arrays work, too)
env_parallel -k echo '${myarray[{}]}' ::: 0 1 2
env_parallel -k -S server echo '${myarray[{}]}' ::: 0 1 2
env_parallel -k --env myarray echo '${myarray[{}]}' ::: 0 1 2
env_parallel -k --env myarray -S server \
  echo '${myarray[{}]}' ::: 0 1 2
```

pdksh

Installation

Put this in $HOME/.profile:

```
source `which env_parallel.pdksh`
```

E.g. by doing:

```
echo 'source `which env_parallel.pdksh`' >> $HOME/.profile
```

Supported use

--env is supported to export only the variable, alias, function, or array with the given name. Multiple **--env**s can be given.

aliases

```
alias myecho="echo aliases";
env_parallel myecho ::: work;
env_parallel -S server myecho ::: work;
env_parallel --env myecho myecho ::: work;
env_parallel --env myecho -S server myecho ::: work
```

functions

```
myfunc() { echo functions $*; };
env_parallel myfunc ::: work;
env_parallel -S server myfunc ::: work;
```

```
env_parallel --env myfunc myfunc ::: work;
env_parallel --env myfunc -S server myfunc ::: work
```

variables

```
myvar=variables;
env_parallel echo "\$myvar" ::: work;
env_parallel -S server echo "\$myvar" ::: work;
env_parallel --env myvar echo "\$myvar" ::: work;
env_parallel --env myvar -S server echo "\$myvar" ::: work
```

arrays

```
myarray=(arrays work, too);
env_parallel -k echo "\${myarray[{}]}" ::: 0 1 2;
env_parallel -k -S server echo "\${myarray[{}]}" ::: 0 1 2;
env_parallel -k --env myarray echo "\${myarray[{}]}" ::: 0 1
2;
env_parallel -k --env myarray -S server \
  echo "\${myarray[{}]}" ::: 0 1 2
```

sh

Installation

Put this in $HOME/.profile:

```
. `which env_parallel.sh`
```

E.g. by doing:

```
echo '. `which env_parallel.sh`' >> $HOME/.profile
```

Supported use

--env is supported to export only the variable, or alias with the given name. Multiple **--env**s can be given.

aliases

```
sh does not support aliases.
```

functions

```
myfunc() { echo functions $*; }
env_parallel myfunc ::: work
env_parallel -S server myfunc ::: work
env_parallel --env myfunc myfunc ::: work
env_parallel --env myfunc -S server myfunc ::: work
```

variables

```
myvar=variables
env_parallel echo '$myvar' ::: work
env_parallel -S server echo '$myvar' ::: work
env_parallel --env myvar echo '$myvar' ::: work
env_parallel --env myvar -S server echo '$myvar' ::: work
```

arrays

```
sh does not support arrays.
```

tcsh

env_parallel for **tcsh** breaks **$PARALLEL**, so do not use **$PARALLEL**.

Installation

Put this in $HOME/.tcshrc:

```
source `which env_parallel.tcsh`
```

E.g. by doing:

```
echo 'source `which env_parallel.tcsh`' >> $HOME/.tcshrc
```

Supported use

--env is supported to export only the variable, alias, or array with the given name. Multiple **--env**s can be given.

aliases

```
alias myecho 'echo aliases'
env_parallel myecho ::: work
env_parallel -S server myecho ::: work
env_parallel --env myecho myecho ::: work
env_parallel --env myecho -S server myecho ::: work
```

functions

Not supported by **tcsh**.

variables

```
set myvar=variables
env_parallel echo '$myvar' ::: work
env_parallel -S server echo '$myvar' ::: work
env_parallel --env myvar echo '$myvar' ::: work
env_parallel --env myvar -S server echo '$myvar' ::: work
```

arrays with no special chars

```
set myarray=(arrays work, too)
env_parallel -k echo \$'{myarray[{}]}' ::: 1 2 3
env_parallel -k -S server echo \$'{myarray[{}]}' ::: 1 2 3
env_parallel -k --env myarray echo \$'{myarray[{}]}' ::: 1 2 3
env_parallel -k --env myarray -S server \
  echo \$'{myarray[{}]}' ::: 1 2 3
```

Zsh

Installation

Put this in $HOME/.zshrc:

```
. `which env_parallel.zsh`
```

E.g. by doing:

```
echo '. `which env_parallel.zsh`' >> $HOME/.zshenv
```

Supported use

--env is supported to export only the variable, alias, function, or array with the given name. Multiple **--env**s can be given.

aliases

```
alias myecho='echo aliases'
env_parallel myecho ::: work
env_parallel -S server myecho ::: work
env_parallel --env myecho myecho ::: work
env_parallel --env myecho -S server myecho ::: work

alias multiline='echo multiline
  echo aliases'
env_parallel multiline ::: work
env_parallel -S server multiline ::: work
env_parallel --env multiline multiline ::: work
env_parallel --env multiline -S server multiline ::: work
```

functions

```
myfunc() { echo functions $*; }
env_parallel myfunc ::: work
env_parallel -S server myfunc ::: work
env_parallel --env myfunc myfunc ::: work
env_parallel --env myfunc -S server myfunc ::: work
```

variables

```
myvar=variables
env_parallel echo '$myvar' ::: work
env_parallel -S server echo '$myvar' ::: work
env_parallel --env myvar echo '$myvar' ::: work
env_parallel --env myvar -S server echo '$myvar' ::: work
```

arrays

```
myarray=(arrays work, too)
env_parallel -k echo '${myarray[{}]}' ::: 1 2 3
env_parallel -k -S server echo '${myarray[{}]}' ::: 1 2 3
env_parallel -k --env myarray echo '${myarray[{}]}' ::: 1 2 3
env_parallel -k --env myarray -S server \
  echo '${myarray[{}]}' ::: 1 2 3
```

EXIT STATUS

Same as GNU **parallel**.

AUTHOR

When using GNU **env_parallel** for a publication please cite:

O. Tange (2011): GNU Parallel - The Command-Line Power Tool, ;login: The USENIX Magazine, February 2011:42-47.

This helps funding further development; and it won't cost you a cent. If you pay 10000 EUR you should feel free to use GNU Parallel without citing.

LICENSE

Documentation license I

Documentation license II

DEPENDENCIES

env_parallel uses GNU **parallel**.

SEE ALSO

parallel(1), **ash**(1), **bash**(1), **csh**(1), **dash**(1), **fish**(1), **ksh**(1), **pdksh**(1) **tcsh**(1), **zsh**(1).