NAME

parset - set shell variables in parallel

SYNOPSIS

parset *variablename* [options for GNU Parallel]

env_parset *variablename* [options for GNU Parallel]

DESCRIPTION

parset is a shell function that puts the output from GNU **parallel** into shell variables.

env_parset is a shell function that puts the output from **env_parallel** into shell variables.

The **parset** and **env_parset** functions are defined as part of **env_parallel**.

If *variablename* is a single variable name, this will be treated as the destination variable. If the variable is defined as an associative array (using **typeset -A myassoc**), this will be used. Otherwise the variable will be made into a normal array.

If *variablename* contains multiple names separated by ',' or space, the names will be the destination variables. The number of names must be at least the number of jobs.

OPTIONS

Same as GNU **parallel**, but they are put *after* the destination variable.

SUPPORTED SHELLS

Bash/Zsh/Ksh/Mksh

Examples

Put output into **myarray**:

```
parset myarray seq 3 ::: 4 5 6
echo "${myarray[1]}"
```

Put output into vars **$seq, $pwd, $ls**:

```
parset "seq pwd ls" ::: "seq 10" pwd ls
echo "$ls"
```

Put output into vars **$seq, $pwd, $ls**:

```
into_vars=(seq pwd ls)
parset "${into_vars[*]}" ::: "seq 10" pwd ls
echo "$ls"
```

Put output into associative array **myassoc** (not supported for mksh):

```
typeset -A myassoc
parset myassoc seq ::: 4 5 ::: 6 7
echo "${myassoc[4 7]}"
```

The commands to run can be an array:

```
cmd=("echo first" "echo '<<joe  \"double  space\"  cartoon>>'" "pwd")
parset data ::: "${cmd[@]}"
echo "${data[1]}"
echo "${data[2]}"
```

parset can read from stdin (standard input) if it is a file:

```
parset res echo < parallel_input_file
```

but **parset** can *not* be part of a pipe. In particular this means it cannot read from a pipe or write to a pipe:

```
seq 10 | parset res echo Does not work
```

but must instead use a tempfile:

```
seq 10 > parallel_input
parset res echo :::: parallel_input
echo "${res[1]}"
echo "${res[9]}"
```

or a FIFO:

```
mkfifo input_fifo
seq 30 > input_fifo &
parset res echo :::: input_fifo
echo "${res[1]}"
echo "${res[29]}"
```

or Bash/Zsh/Ksh process substitution:

```
parset res echo :::: <(seq 100)
echo "${res[1]}"
echo "${res[99]}"
```

Installation

Put this in the relevant **$HOME/.bashrc** or **$HOME/.zshenv** or **$HOME/.kshrc**:

```
. `which env_parallel.bash`
. `which env_parallel.zsh`
source `which env_parallel.ksh`
```

E.g. by doing:

```
echo '. `which env_parallel.bash`' >> $HOME/.bashrc
echo '. `which env_parallel.zsh`' >> $HOME/.zshenv
echo 'source `which env_parallel.ksh`' >> $HOME/.kshrc
```

or by doing:

```
env_parallel --install
```

ash/dash (FreeBSD's /bin/sh)

Examples

ash does not support arrays.

Put output into vars **$seq, $pwd, $ls**:

```
parset "seq pwd ls" ::: "seq 10" pwd ls
echo "$ls"
```

parset can read from stdin (standard input) if it is a file:

```
parset res1,res2,res3 echo < parallel_input_file
```

but **parset** can not be part of a pipe. In particular this means it cannot read from a pipe or write to a pipe:

```
seq 3 | parset res1,res2,res3 echo Does not work
```

but must instead use a tempfile:

```
seq 3 > parallel_input
parset res1,res2,res3 echo :::: parallel_input
echo "$res1"
echo "$res2"
echo "$res3"
```

or a FIFO:

```
mkfifo input_fifo
seq 3 > input_fifo &
parset res1,res2,res3 echo :::: input_fifo
echo "$res1"
echo "$res2"
echo "$res3"
```

Installation

Put the relevant one of these into **$HOME/.profile**:

```
. `which env_parallel.sh`
. `which env_parallel.ash`
. `which env_parallel.dash`
```

E.g. by doing:

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

or by doing:

```
env_parallel --install
```

EXIT STATUS

Same as GNU **parallel**.

AUTHOR

When using GNU **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

parset uses GNU **parallel**.

SEE ALSO

parallel(1), **env_parallel**(1), **bash**(1).