SPACE APPLICATION NOTE ABOUT MKPR PATH USAGE

S. de Graaf

Circuits and Systems Group
Faculty of Electrical Engineering
Delft University of Technology
The Netherlands

Report ET-CAS 00-07
December 1, 2000

Last revision: December 16, 2003.

1. INTRODUCTION

Where is the real location of the process directory (data)?
This is an important question, when using the *mkpr* program.

When the process directory is an absolute path, than there is no problem. When the process specification is a process name or a process id, which can be found in the process list, than there is also no problem.

But, what happens, when the process directory is specified with a relative path? With option **-p** gives this no problem, because the path is relative against the cwd.

> **NOTE:**
> I have made the use of CWD by *mkpr* not possible. However, when the env.var. CWD was used, than it replaces the cwd. Than you make the new project (when relative), relative against the CWD (in place of cwd). But in that case are also the given relative paths for option **-u** and **-p** relative against CWD. Therefor, i have chosen, not to implement that possibility. It makes it too complicated.

But, when the inheritance find place out of another project (option **-u**) and the found process path in the ".dmrc" file is relative?

Then there are also two possibilities:
1) project path is relative specified (against the cwd), or
2) project path is absolute specified.

The question in that case is, what is the relation between the to make project path (p_new) and the to use project path (p_use)? Above possibilities must be taken into account both for p_new and p_use. With relation, i mean, the position in the hierarchical file system, which they have against each other.

Four possibilities:

```
      p_new p_use
-------------------------
1)    rel.  rel.  both paths against "." (cwd)
2)    rel.  abs.
3)    abs.  rel.
4)    abs.  abs.  both paths against "/" (root)
```

ad 1) By a relative position against cwd, directly a
relative position against each other is chosen.
ad 2-4) By an absolute position is ‘‘hard’’ given, where the directory is.
Than, their relation must be chosen against the root.
Thus look, which part of the path they have in common.

NOTE:
The relation they have can be maskered, when the given absolute (or relative) path contains symbolic links.

NOTE:
Directory cwd (or CWD) is only used, when you specify a relative path.

Thus, the paths must be changed into their real absolute paths. Like the *pwd* program does and functions chdir()/getcwd().

Two absolute paths have always, in each case, the root directory '/' in common. One path can be a sub-directory (or sub-sub-dir) of the other path. In that case, they have both the first part of the path fully in common.

1.1 Situation 1

When dir_a/dir_b is a sub-directory of dir_b/dir_a, than is a relative path to each other (almost) always the shortest solution. Examples:

```
(a) /u/13/13/simon/a
(b) /u/13/13/simon/a/b
To chdir from (a) to (b), do: cd b
To chdir from (b) to (a), do: cd ..

(a) /u/13/13/simon/a
(b) /u/13/13/simon/a/c/b
To chdir from (a) to (b), do: cd c/b
To chdir from (b) to (a), do: cd ../..

(a) /
(b) /b
To chdir from (a) to (b), do: cd b     (abs.: cd /b )
To chdir from (b) to (a), do: cd ..    (abs.: cd /  )

(a) /
(b) /c/b
To chdir from (a) to (b), do: cd c/b   (abs.: cd /c/b )
To chdir from (b) to (a), do: cd ../.. (abs.: cd /    )
```

Going downward, a relative path is always the shortest path. Going upward, a relative path (using: ..) is often the shortest path, but not always (depending on the length of the shortest absolute path).

1.2 Situation 2

When dir_a and dir_b have their parent directory in common. Example:

```
(a) /u/13/13/simon/a
(b) /u/13/13/simon/b
To chdir from (a) to (b), do: cd ../b
To chdir from (b) to (a), do: cd ../a
```

1.3 Situation 3

When dir_a and dir_b don’t have directly their parent in common. Example:

```
(a) /u/13/13/simon/a
(b) /u/13/13/simon/c/b
To chdir from (a) to (b), do: cd ../c/b
To chdir from (b) to (a), do: cd ../../a
```

Thus, you see, for a relative chdir to each other you must first go upstairs, to a common point and than you go downstairs. Note, when two paths have only the root in common, than it is not wise to make use of a relative given path.

Thus, the question is, when shall we use relative and when absolute? If possible, when we start with a relative specified process directory path, we want to keep that path relative.

But what, when the inheritance find place out of another project (option **-u**) and the found process path was in the ".dmrc" file. This relative process path (p_proc) has a relation with p_use, but does not need to have a relation with p_new!

We must make one of the following choices:

1. Choice absolute, when they have only the root in common.
2. Choice relative, when the path part to the common point shorter looks to be, than the absolute path part they have in common. Else choice absolute.