Label quoting snippets

One doc to unite all the proposed quoting and in the dark bind them.

Author(s): dmarting@bazel.build, philwo@bazel.build
Updates:

- 2017-08-30 Update - dmarting, philwo
- 2017-03-17 Initial version - dmarting@

This document should be used to discuss proposal for label quoting. It completes previous document about supporting arbitrary characters in filenames for Bazel.

Special labels

This table present the package and target part of the label that we try to use:

	Package	Target
A	my$(location funny)package	my$(locations funny)target
B	with"quotes"and'\nslashes`${and more}%	with"quotes"and'\nslashes`${and more} %
C	pkg:with:colons	target:with:colons

Showcases

Case 1. Genrule

```
genrule(
    name = "<target>",
    outs = ["<target>.txt", "label.txt"],
    srcs = ["//<package>:<target>"],
    cmd = "\n".join([
         "cp '$(location //<package>:<target>)' '$(location <target>.txt)'",
         "echo '//<package>:<target>' >'$(location label.txt)'",
    ]),
)
```

Case 2. Skylark macro

```
def my_macro(name, src, outs):
  label = Label(MY_LABEL)
  label = str(label)
  native.genrule(
    srcs = [src, label],
    outs = outs,
    cmd = "'$(location %s)' '$(location %s)' %s" % (label, src, shell_quote(src)))
```

Where MY_LABEL is the label with special characters

Proposals

Proposal 1. Change the label parser

The proposal is to have the label parser handle quoting. The current proposal use percent quoting and for the sake of simplicity, this document will simply use URL percent encoding when needed. The parser itself would just require ':' to be encoded, but would understand everything.

Application

Case 1. Label A

`//my$(location funny)package:my$(location funny)target` can be encoded as
`//my%44(location funny%41package:my%44(location funny%41target`

```
genrule(
    name = "my$(location funny)target",
    outs = ["my$(location funny)target.txt", "label.txt"],
    srcs = ["//my$(location funny)package:my$(location funny)target"],
    cmd = "\n".join([
         "cp '$(location //my%44(location funny%41package:my%44(location
funny%41target)' '$(location my%44(location funny%41target.txt)'",
         "echo '//my%44(location funny%41package:my%44(location funny%41target'
>'$(location label.txt)'",
    ]),
)
```

Case 1. Label B

No need to encode the label, we can use python syntax most of the case (the choice of the user to encode or not):

```
genrule(
    name = "with\"quotes\"and'\\nslashes`${and more}\n%25",
    outs = ["with\"quotes\"and'\\nslashes`${and more}\n%25.txt", "label.txt"],
    srcs = ["//with\"quotes\"and'\\nslashes`${and
more}%25:with\"quotes\"and'\\nslashes`${and more}\n%25"],
    cmd = "\n".join([
         "cp '$(location //with\"quotes\"and'\\nslashes`${and
more}%25:with\"quotes\"and'\\nslashes`${and more}\n%25)' '$(location
with\"quotes\"and'\\nslashes`${and more}\n%25.txt)'",
         "echo 'with\"quotes\"and%2C\\nslashes`${and
more}%25:with\"quotes\"and%2C\\nslashes`${and more}\n%25' >'$(location
label.txt)'",
    ]),
)
```

Case 1. Label C

This one needs encoding of the colon for the label parser:
`\\pkg%3Awith%3Acolons:label%3Awith%3Acolons`

```
genrule(
    name = "label:with:colons",
    outs = ["label:with:colons.txt", "label.txt"],
    srcs = ["//pkg%3Awith%3Acolons:label%3Awith%3Acolons"],
    cmd = "\n".join([
         "cp '$(location //pkg%3Awith%3Acolons:label%3Awith%3Acolons)' '$(location
label%3Awith%3Acolons.txt)'",
         "echo
'//pkg%3Awith%3Acolons:label%3Awith%3Acolonswith\"quotes\"and%2C\\nslashes`${and
```

```
more}%:with\"quotes\"and%2C\\nslashes`${and more}\n%' >'$(location label.txt)'",
    ]),
)
```

Case 2. Label A, B and C

```
def my_macro(name, src, outs):
  label = Label(MY_LABEL)
  native.genrule(
    srcs = [src, MY_LABEL],
    outs = outs,
    cmd = "'$(location %s)' '$(location %s)' %s" % (Label(MY_LABEL), src,
shell_quote(src)))
```

By having label string conversion quotes the following characters: '$', '\n', ')', '%', ':' and newline.

Proposal 2. philwo

%% → %
%: → :