Q: The following code snippet is excerpted from a Bourne shell script which attempts to determine how to invoke the C preprocessor on any given UNIX platform. The script creates a file called testcpp.c and runs the C preprocessor on that file redirecting the output to testcpp.out. When I run this script under NEXTSTEP it incorrectly inserts a space in the output file "abc +xyz". This causes the test which confirms "cc -E" as the correct invocation to fail. Is this a bug in your C preprocessor?

```
: see how we invoke the C preprocessor
echo " "
echo "Checking to see how your C preprocessor is invoked..."
cat <<'EOT' >testcpp.c
#define ABC abc
#define XYZ xyz
ABC+XYZ
```

```
EOT
echo 'Maybe "cc -E" will work...'
cc -E testcpp.c >testcpp.out 2>&1
if $contains 'abc+xyz' testcpp.out >/dev/null 2>&1 ; then
   echo "Yup, it does."
   cpp='cc -E'
else
   echo 'Nope...maybe "cc -P" will work...'
   cc -P testcpp.c >testcpp.out 2>&1
   if $contains 'abc+xyz' testcpp.out >/dev/null 2>&1 ; then
            echo "Yup, that does."
```

A: The space inserted before the "+xyz" follows the behavior specified by ANSI C. If you want the old behavior of standard UNIX compilers, use the -traditional flag. The man page on cc(1) shows a

large number of useful compiler flags in addition to -traditional. The flags -bsd and -fwritable-strings may also be needed when compiling old UNIX programs.

QA594

Valid for 2.0, 3.0