```
##Aim:
###To Write a C/C++ POSIX compliant program to check the following
limits: <br>(i) No. of clock ticks<br>(ii) Max. no. of child
processes<br>(iii) Max. path length<br>(iv) Max. no. of characters
in a file name<br>(v) Max. no. of open files/ process



##Theory:

###**sysconf** - *Get configuration information at runtime*

<blockquote>

SYNOPSIS
<blockquote>

       <pre><code>#include &lt;unistd.h&gt;
long sysconf(int name);</code></pre>
</blockquote>


DESCRIPTION
<blockquote>

<pre><p>POSIX  allows  an  application to test at compile or run
time whether certain options are supported, or what the value is of
certain configurable constants or limits.At compile time this is
done by including &lt;unistd.h&gt; and/or &lt;limits.h&gt; and
testing the value of certain macros.</p>
<p>At run time, one can ask for numerical values using the present
function sysconf(). One can ask for numerical  values  that may
depend  on  the  file  system a file is in using the calls
fpathconf(3) and pathconf(3). One can ask for string values using
confstr(3). The values obtained from these functions are system
configuration constants. They do not change during the lifetime of a
process.</p></pre>


<blockquote>

<pre>clock ticks - _SC_CLK_TCK
              The number of clock ticks per second.  The
corresponding variable is obsolete.  It  was  of  course  called
CLK_TCK.

CHILD_MAX - _SC_CHILD_MAX
              The max number of simultaneous processes per user ID.
Must not be less than _POSIX_CHILD_MAX (25).

OPEN_MAX - _SC_OPEN_MAX
              The maximum number of files that a process can have
open at any time.  Must not be less than _POSIX_OPEN_MAX (20).</pre>
```

```
</blockquote>


</blockquote>

</blockquote>

###fpathconf, pathconf - *Get configuration values for files*

<blockquote>

SYNOPSIS

<blockquote>

<pre><code>#include &lt;unistd.h&gt;
long fpathconf(int fd, int name);
long pathconf(char *path, int name);</code></pre>

</blockquote>

DESCRIPTION
<blockquote>

<pre>fpathconf() gets a value for the configuration option name for
the open file descriptor fd.

pathconf() gets a value for configuration option name for the
filename path.

The corresponding macros defined in &lt;unistd.h&gt; are minimum
values; if an application wants to take advantage of values which
may change, a call to fpathconf() or pathconf() can be made, which
may yield more liberal results.</pre>

<blockquote>

<pre>_PC_PATH_MAX
              returns the maximum length of a relative pathname when
path or fd is the current working directory.  The  corresponding
macro is _POSIX_PATH_MAX.

_PC_NAME_MAX
              returns the maximum length of a filename in the
directory path or fd that the process is allowed to create.  The
corresponding macro is _POSIX_NAME_MAX.</pre>

</blockquote>

</blockquote>
</blockquote>
##Code:
```

```
<pre><code>#define _POSIX_SOURCE
#define _POSIX_C_SOURCE 199309L
#include "iostream"
#include &lt;unistd.h&gt;
using namespace std;
int main()
{
        cout&lt;&lt;"No of clock
ticks:"&lt;&lt;sysconf(_SC_CLK_TCK)&lt;&lt;endl;
        cout&lt;&lt;"Maximum no of child
processes:"&lt;&lt;sysconf(_SC_CHILD_MAX)&lt;&lt;endl;
        cout&lt;&lt;"Maximum path
length:"&lt;&lt;pathconf("/",_PC_PATH_MAX)&lt;&lt;endl;
        cout&lt;&lt;"Maximum characters in a file
name:"&lt;&lt;pathconf("/",_PC_NAME_MAX)&lt;&lt;endl;
        cout&lt;&lt;"Maximum no of open
files:"&lt;&lt;sysconf(_SC_OPEN_MAX)&lt;&lt;endl;
        return 0;
}
</code></pre>

##Output:

*Commands for execution:-*
<ul>
    <li> Open a terminal.</li>
    <li> Change directory to the file location in the terminal.</li>
    <li> Run g++ usp01.c -o usp01.out in the terminal.</li>
    <li> If no errors, run ./usp01.out</li>


##Screenshots:

 ![not available](usp-lab-01.png "usp01 screenshot")
```