Enabling Gperf tool for memory leak

In-progress

Gperf-tool

- Gperftools is a set of tools for performance profiling and memory checking

Build steps

Follow the link for RPi build steps- RDK-B R-Pi Build guide

Installation:

Once after successful repo steps executed, below changes are needed to install gperf

In the image recipe(i.e rdkb-generic-broadband-image.bbappend)

```
IMAGE_INSTALL_append = " gperftools"
```

To avoid incompatible license errors with binutils(dependency for gperftools),
Add below changes in $YOCTO_BUILD/conf/local.conf:

```
WHITELIST_GPL-3.0 += " binutils"
INCOMPATIBLE_LICENSE = "GPL-3.0 LGPL-3.0 AGPL-3.0"
```

Execution

Binaries generated and Responsible binaries for execution:

```
$ pprof
$ pprof-symbolize
```

command to be used,

```
/usr/bin/pprof [options] <program> <profiles>
```

Usage:

```
root@RaspberryPi-Gateway:~# pprof
Did not specify program
/usr/bin/pprof [options] <program> <profiles>
<profiles> is a space separated list of profile names.
/usr/bin/pprof [options] <symbolized-profiles>
<symbolized-profiles> is a list of profile files where each file contains
the necessary symbol mappings as well as profile data (likely generated
with --raw).
/usr/bin/pprof [options] <profile>
<profile> is a remote form. Symbols are obtained from host:port/pprof/symbol
Each name can be:
/path/to/profile - a path to a profile file
host:port[/<service>] - a location of a service to get profile from
The /<service> can be /pprof/heap, /pprof/profile, /pprof/pmuprofile,
/pprof/growth, /pprof/contention, /pprof/wall,
```

```
/pprof/censusprofile(?:\?.*)?, or /pprof/filteredprofile.
For instance:
/usr/bin/pprof http://myserver.com:80/pprof/heap
If /<service> is omitted, the service defaults to /pprof/profile (cpu profiling).
/usr/bin/pprof --symbols <program>
Maps addresses to symbol names. In this mode, stdin should be a
list of library mappings, in the same format as is found in the heap-
and cpu-profile files (this loosely matches that of /proc/self/maps
on linux), followed by a list of hex addresses to map, one per line.
For more help with querying remote servers, including how to add the
necessary server-side support code, see this filename (or one like it):
/usr/doc/gperftools-2.0/pprof_remote_servers.html
Options:
--cum Sort by cumulative data
--base=<base> Subtract <base> from <profile> before display
--interactive Run in interactive mode (interactive "help" gives help) [default]
--seconds=<n> Length of time for dynamic profiles [default=30 secs]
--add_lib=<file> Read additional symbols and line info from the given library
--lib_prefix=<dir> Comma separated list of library path prefixes
--no_strip_temp Do not strip template arguments from function names
Reporting Granularity:
--addresses Report at address level
--lines Report at source line level
--functions Report at function level [default]
--files Report at source file level
Output type:
--text Generate text report
--stacks Generate stack traces similar to the heap profiler (requires --text)
--callgrind Generate callgrind format to stdout
--gv Generate Postscript and display
--evince Generate PDF and display
--web Generate SVG and display
--list=<regexp> Generate source listing of matching routines
--disasm=<regexp> Generate disassembly of matching routines
--symbols Print demangled symbol names found at given addresses
--dot Generate DOT file to stdout
--ps Generate Postscript to stdout
--pdf Generate PDF to stdout
--svg Generate SVG to stdout
--gif Generate GIF to stdout
--raw Generate symbolized pprof data (useful with remote fetch)
--collapsed Generate collapsed stacks for building flame graphs
(see http://www.brendangregg.com/flamegraphs.html)
Heap-Profile Options:
--inuse_space Display in-use (mega)bytes [default]
--inuse_objects Display in-use objects
--alloc_space Display allocated (mega)bytes
--alloc_objects Display allocated objects
--show_bytes Display space in bytes
--drop_negative Ignore negative differences
Contention-profile options:
--total_delay Display total delay at each region [default]
--contentions Display number of delays at each region
--mean_delay Display mean delay at each region
Call-graph Options:
--nodecount=<n> Show at most so many nodes [default=80]
--nodefraction=<f> Hide nodes below <f>*total [default=.005]
--edgefraction=<f> Hide edges below <f>*total [default=.001]
--maxdegree=<n> Max incoming/outgoing edges per node [default=8]
--focus=<regexp> Focus on nodes matching <regexp>
--ignore=<regexp> Ignore nodes matching <regexp>
--scale=<n> Set GV scaling [default=0]
--heapcheck Make nodes with non-0 object counts
(i.e. direct leak generators) more visible
Miscellaneous:
--no-auto-signal-frm Automatically drop 2nd frame that is always same (cpu-only)
(assuming that it is artifact of bad stack captures
which include signal handler frames)
--show_addresses Always show addresses when applicable
--tools=<prefix or binary:fullpath>[,...] $PATH for object tool pathnames
--test Run unit tests
```

```
--help This message
--version Version information
Environment Variables:
PPROF_TMPDIR Profiles directory. Defaults to $HOME/pprof
PPROF_TOOLS Prefix for object tools pathnames
Examples:
/usr/bin/pprof /bin/ls ls.prof
Enters "interactive" mode
/usr/bin/pprof --text /bin/ls ls.prof
Outputs one line per procedure
/usr/bin/pprof --web /bin/ls ls.prof
Displays annotated call-graph in web browser
/usr/bin/pprof --gv /bin/ls ls.prof
Displays annotated call-graph via 'gv'
/usr/bin/pprof --gv --focus=Mutex /bin/ls ls.prof
Restricts to code paths including a .*Mutex.* entry
/usr/bin/pprof --gv --focus=Mutex --ignore=string /bin/ls ls.prof
Code paths including Mutex but not string
/usr/bin/pprof --list=getdir /bin/ls ls.prof
(Per-line) annotated source listing for getdir()
/usr/bin/pprof --disasm=getdir /bin/ls ls.prof
(Per-PC) annotated disassembly for getdir()
/usr/bin/pprof http://localhost:1234/
Enters "interactive" mode
/usr/bin/pprof --text localhost:1234
Outputs one line per procedure for localhost:1234
/usr/bin/pprof --raw localhost:1234 > ./local.raw
/usr/bin/pprof --text ./local.raw
Fetches a remote profile for later analysis and then analyzes it in text mode.
```

Errors during execution

```
root@RaspberryPi-Gateway:~# pprof --text /usr/bin/CcspPandMSsp ccsp.prof
Using local file /usr/bin/CcspPandMSsp.
Argument "MSWin32" isn't numeric in numeric eq (==) at /usr/bin/pprof line 5047.
Argument "linux" isn't numeric in numeric eq (==) at /usr/bin/pprof line 5047.
Gathering CPU profile from http://ccsp.prof/pprof/profile?seconds=30 for 30 seconds to
/home/root/pprof/CcspPandMSsp.1668162550.ccsp.prof
Be patient...
Failed to get profile: curl -s --max-time 90 'http://ccsp.prof/pprof/profile?seconds=30' > /home/root/pprof/.
tmp.CcspPandMSsp.1668162550.ccsp.prof: No such file or directory
```

Limitations

- Need to solve the error with pprof and get the results that helps in analysis.