Coredump Library Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

1 Library Overview

Scope of This Document

The Coredump library is a library for programs to write arbitrary information as part of the core file. In addition to the basic information included in the core file, information useful in debugging will be written, allowing confirmation of the status at the time an exception occurs.

Purpose and Features

By using the Coredump library, arbitrary data can be embedded as part of the core file when an application crashes. By a program registering a callback function (the following core dump handler) called when an application crashes and by calling the API for writing the user data in the core dump handler, the user data to be inserted into the core file will be defined. Through this, it will be possible to embed uniquely-formatted data useful for problem analysis into the core file according to the status of the application when the crash occurred.

In addition, by using the Coredump library in combination with the core file upload feature, a system can be constructed for collecting and analyzing application-unique information in addition to the basic debug information included in the core file. Since the core dump and upload feature are provided by the system software, the implementation of the core dump handler and setting the web server are the only operations required by developers.

Main Features

The main features offered by the Coredump library are as follows:

- Feature for registering the callback function called when an application crashes in a program
- Feature for embedding arbitrary data in part of a core file

Files

The files required to use the Coredump library are as follows.

File Name	Description
coredump.h	Header file
libSceCoredump_stub.a	Stub library file

Sample Program

The following program is provided as a Coredump library sample program for reference purposes.

sample_code/developer_tools/api_coredump/

This shows the basic method for using Coredump library.

2 Using the Library

Basic Usage Procedure

(1) Define the core dump handler

Define a `SceCoredumpHandler` type function called when an application crashes as the core dump handler. Call `sceCoredumpWriteUserData()` in the core dump handler and specify the user data to write to the core file. The user data can be any format. `sceCoredumpWriteUserData()` can be called multiple times in the core dump handler, with data written to the core file in the order of execution.

(2) Register the core dump handler

Call `sceCoredumpRegisterCoredumpHandler()` to register the core dump handler address and the stack size of thread that will execute the core dump handler. Memory at the size specified at the time of the core dump handler registration will be reserved from the application's memory budget to the system.

When an application crashes, a user thread will be generated, the registered core dump handler will be called, and the user data will be written to the core file.

Main APIs used for basic processing

API	Description
`SceCoredumpHandler`	Function type definition to register as the core dump handler
`sceCoredumpRegisterCoredumpHandler()`	Registers the core dump handler
`sceCoredumpWriteUserData()`	Performs the user data write
`sceCoredumpUnregisterCoredumpHandler()`	Unregisters the core dump handler

User Data Read Method

By using the "extract-userdata" command of psp2ctrl (one of the command line utilities), the user data can be extracted as a file from the core file specified with a command line argument.

Notes on Usage

Restriction on the execution time for the core dump handler

Make sure the processing completes for the core dump handler within three seconds after execution starts. If the execution time exceeds three seconds, the thread executing the core dump handler will be forcibly terminated.

Restriction on the system call calling in the core dump handler

The core dump handler is executed only for the purpose of embedding user data in a core file. All system calls and libraries provided by SCE for calling system calls cannot be used with the exception of user data writing API (`sceCoredumpWriteUserData()`) and `sceKernelExitThread()`. libc is included in the libraries for which this restriction applies, but usage of libc functions in the core dump handler is possible when limited to the following.

[stdio.h]

`snprintf`, `sprintf`, `sscanf`, `vsnprintf`, `vsprintf`, `vsscanf`

[stdlib.h]

`abs`, `atof`, `atoff`, `atoi`, `atol`, `atoll`, `bsearch`, `div`, `getargc`, `getargv`, `labs`, `llabs`, `ldiv`, `lldiv`, `mblen`, `mbstowcs`, `mbtowc`, `qsort`, `rand`, `rand_r`, `srand`, `strtod`, `strtof`, `strtol`, `strtold`, `strtoll`, `strtoul`, `strtoull`, `wcstombs`, `wctomb`

[string.h]

`memchr`, `memcmp`, `memcpy`, `memmove`, `memset`, `strcat`, `strchr`, `strcmp`, `strcoll`, `strcpy`, `strcspn`, `strerror`, `strlen`, `strncat`, `strncmp`, `strncpy`, `strpbrk`, `strrchr`, `strspn`, `strstr`, `strtok`, `strtok_r`, `strxfrm`, `strcasecmp`, `strncasecmp`

[math.h]

`abs`, `acos`, `acosf`, `acosh`, `acoshf`, `acoshl`, `acosl`, `asin`, `asinf`, `asinh`, `asinhf`, `asinhl`, `asinl`, `atan`, `atan2`, `atan2f`, `atan2l`, `atanf`, `atanh`, `atanhf`, `atanhl`, `atanl`, `cbrt`, `cbrtf`, `cbrtl`, `ceil`, `ceilf`, `ceill`, `copysign`, `copysignf`, `copysignl`, `cos`, `cosf`, `cosh`, `coshf`, `coshl`, `cosl`, `erf`, `erfc`, `erfcf`, `erfcl`, `erff`, `erfl`, `exp`, `exp2`, `exp2f`, `exp2l`, `expf`, `expl`, `expm1`, `expm1f`, `expm1l`, `fabs`, `fabsf`, `fabsl`, `fdim`, `fdimf`, `fdiml`, `floor`, `floorf`, `floorl`, `fma`, `fmaf`, `fmal`, `fmax`, `fmaxf`, `fmaxl`, `fmin`, `fminf`, `fminl`, `fmod`, `fmodf`, `fmodl`, `fpclassify`, `frexp`, `frexpf`, `frexpl`, `hypot`, `hypotf`, `hypotl`, `ilogb`, `ilogbf`, `ilogbl`, `isfinite`, `isgreater`, `isgreaterequal`, `isinf`, `isless`, `islessequal`, `islessgreater`, `isnan`, `isnormal`, `isunordered`, `ldexp`, `ldexpf`, `ldexpl`, `lgamma`, `lgammaf`, `lgammal`, `llrint`, `llrintf`, `llrintl`, `llround`, `llroundf`, `llroundl`, `log`, `log10`, `log10f`, `log10l`, `log1p`, `log1pf`, `log1pl`, `log2`, `log2f`, `log2l`, `logb`, `logbf`, `logbl`, `logf`, `logl`, `lrint`, `lrintf`, `lrintl`, `lround`, `lroundf`, `lroundl`, `modf`, `modff`, `modfl`, `nan`, `nanf`, `nanl`, `nearbyint`, `nearbyintf`, `nearbyintl`, `nextafter`, `nextafterf`, `nextafterl`, `nexttoward`, `nexttowardf`, `nexttowardl`, `pow`, `powf`, `powl`, `remainder`, `remainderf`, `remainderl`, `remquo`, `remquof`, `remquol`, `rint`, `rintf`, `rintl`, `round`, `roundf`, `roundl`, `scalbln`, `scalblnf`, `scalblnl`, `scalbn`, `scalbnf`, `scalbnl`, `signbit`, `sin`, `sinf`, `sinh`, `sinhf`, `sinhl`, `sinl`, `sqrt`, `sqrtf`, `sqrtl`, `tan`, `tanf`, `tanh`, `tanhf`, `tanhl`, `tanl`, `tgamma`, `tgammaf`, `tgammal`, `trunc`, `truncf`, `truncl`

Restriction on Language Features in a Core Dump Handler

In accordance with the system call restriction, the following language features cannot be used in a core dump handler.

- C++ exceptions
- Dynamic casting
- `typeid`
- Static objects in functions

Restriction on user data writes

The maximum size for user data is 16384 bytes. In addition, the user data write API that can be used is limited to the inside of the core dump hander.

Restriction when the Release Check Mode is Development Mode

When the **Release Check Mode** is **Development Mode** and a core file has been generated with the following methods, the core dump handler will not be executed.

- Selecting **Save as** from the **Debug** menu of Visual Studio
- Using **pdump** of psp2ctrl
- Using the Target Manager API `InitiateCoredump()`