DEOS-FY2013-SP-01J

JST-CREST

研究領域

「実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム」

DEOS プロジェクト

D-RE API サンプルプログラム解説書

Version 1.02

2013/09/01

DEOS 研究開発センター

科学技術振興機構
Japan Science and Technology Agency

目次

変更履歴

Version	変更内容	変更者	日付
1.01	誤表記を修正	Dependable Embedded OS R&D Center	2013.Jul.15
1.02	誤表記、記載漏れを修正	Dependable Embedded OS R&D Center	2013.Sep.1

1　コンテナ API をコールするサンプルプログラム

1.1　アプリケーションコンテナの API をコールするサンプルプログラム

1.1.1サンプルプログラムの使用方法

libdre0-dev をインストールすると、以下のパスに本 API を使用したサンプルプログラムのソースコードがインストールされます。

/usr/share/doc/libdre0-dev/sample/app

上記のパスで make コマンドを実行すると、サンプルプログラムがコンパイルできます。

1.1.2サンプルプログラムの解説

コンテナ API には、以下のサンプルプログラムが付属します。

dre_app_bind_pid.c
アプリケーションコンテナ上のコントロールグループにプロセスをバインドするサンプル

dre_app_commit_snapshot.c
アプリケーションコンテナのスナップショットを使用して、ベースイメージを新規作成するサンプル

dre_app_create_cgroup.c
アプリケーションコンテナ上にコントロールグループを作成するサンプル

dre_app_create_container.c
アプリケーションコンテナを作成するサンプル

dre_app_destroy_cgroup.c
アプリケーションコンテナ上のコントロールグループを破棄するサンプル

dre_app_destroy_container.c
アプリケーションコンテナを破棄するサンプル

dre_app_get_base_list.c
存在するベースイメージを列挙するサンプル

dre_app_get_container_list.c
存在するアプリケーションコンテナを列挙するサンプル

dre_app_get_registered_pid_list.c
/proc/daware に登録されたプロセスの PID を列挙するサンプル

dre_app_get_resource.c
アプリケーションコンテナが使用するリソースの情報を取得するサンプル

dre_app_get_snapshot_list.c
アプリケーションコンテナのスナップショットを列挙するサンプル

dre_app_get_status
アプリケーションコンテナの状態を取得するサンプル

dre_app_load_snapshot.c
アプリケーションコンテナをスナップショットの状態に復元するサンプル

dre_app_remove_base.c
ベースイメージを削除するサンプル

dre_app_remove_snapshot.c
アプリケーションコンテナのスナップショットを削除するサンプル

dre_app_resume.c

一時停止中のアプリケーションコンテナの動作を再開するサンプル

dre_app_save_snapshot.c

アプリケーションコンテナのスナップショットを作成するサンプル

dre_app_set_resource.c

アプリケーションコンテナが使用するリソースに制限をかけるサンプル

dre_app_start.c

アプリケーションコンテナの動作を開始するサンプル

dre_app_stop.c

アプリケーションコンテナの動作を停止するサンプル

dre_app_suspend.c

動作中のアプリケーションコンテナを一時停止するサンプル

dre_app_unbind_pid.c

アプリケーションコンテナ上のコントロールグループからプロセスをアンバインドするサンプル

dre_app_unset_resource.c

アプリケーションコンテナのリソース制限を解除するサンプル

1.1.2.1 dre_app_bind_pid.c

概要

コントロールグループにプロセスをバインドするサンプル(“dre-app bind [-n <name>] -g <cgroup> -pid <pid>”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
10          char *cgroup = NULL;
            ・・・(略)・・・
27          /* open handler */
29          h = dre_app_open(name);
30          if(!h) {
31                  result = DRE_ERR_NULL_POINTER;
32                  goto error;
34          }
35
36          /* register pid */
37          result = dre_app_register_pid(h, pid);
38          if(result) goto error;
39
40          /* bind to control group */
41          result = dre_app_bind_pid(h, cgroup, pid);
42          if(result) goto error;
```

```
43
44        /* unregister pid */
45        result = dre_app_unregister_pid(h, pid);
46        if(result) goto error;
47
48        /* close handler */
50        result = dre_app_close(h);
51        if(result) goto error;
          ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_bind_pid.c を参照。)

プログラムの引数でコンテナ名<name>を指定した場合、29 行目の dre_app_open()で、<name>のハンドルを取得します。37 行目の dre_app_register_pid()で、ホスト OS で割り当てられたプロセス ID(PID)を、アプリケーションコンテナ上で割り当てられたプロセス ID (VPID)と紐付けます。41 行目の dre_app_bind_pid()で、<pid>をコントロールグループ<cgroup>にバインドします。この時、<cgroup>は、<name>が指定された場合は<name>上のものが使用され、<name>を省略した場合はホスト OS 上のものが使用されます。45 行目の dre_app_unregister_pid()で、PID と VPID の紐付けを解除します。<name>を指定した場合、50 行目の dre_app_close()で<name>のハンドルを解放します。

使用方法

dre_app_bind_pid　-g <cgroup>　--pid <pid>　[-n <name>]

引数

<cgroup>	コントロールグループ名
<pid>	プロセス ID
<name>	コンテナ名(オプション)

1.1.2.2　dre_app_commit_snapshot.c

概要

アプリケーションコンテナのスナップショットを使用して、ベースイメージを新規作成するサンプル
(“dre-app commit -n <name> -t <tag> -b <base_image>”に相当)

ソースコード解説

```
3   #include “dre_app.h”
4
5   int main(int argc, char **argv)
6   {
7         dre_return_t result;
8         dre_app_handle *h = NULL;
9         char *name = NULL;
10        char *tag = NULL;
11        char *base = NULL;
          ・・・(略)・・・
27        /* open handler */
28        h = dre_app_open(name);
29        if(!h) {
```

```
30              result = DRE_ERR_NULL_POINTER;
31              goto error;
32      }
33
34      /* commit snapshot */
35      result = dre_app_commit_snapshot(h, snapshot, base);
36      if(result) goto error;
37
38      /* close handler */
39      result = dre_app_close(h);
40      if(result) goto error;
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_commit_snapshot.c を参照。)

28 行目の dre_app_open()で、コンテナ<name>のハンドルを取得します。35 行目の dre_app_commit_snapshot()で、<name>のスナップショット<tag>を使用してベースイメージ<base_image>を作成します。39 行目の dre_app_close()で、<name>のハンドルを解放します。

使用方法

dre_app_commit_snapshot -n <name>　-t <tag>　-b <base_image>

引数

<name>	コンテナ名
<tag>	スナップショット名
<base_image>	作成するイメージ名

1.1.2.3　dre_app_create_cgroup.c

概要

コントロールグループを作成するサンプル(“dre-app create [-n <name>] -g <cgroup>”に相当)

ソースコード解説

```
3   #include “dre_app.h”
4
5   int main(int argc, char **argv)
6   {
7       dre_return_t result:
8       dre_app_handle *h = NULL;
9       char *name = NULL;
10      char *cgroup = NULL;
        ・・・(略)・・・
24      /* open handler */
25      if(name) {
26          h = dre_app_open(name);
27          if(!h) {
28              result = DRE_ERR_NULL_POINTER;
29              goto error;
```

```
30              }
31      }
32
33      /* create control group */
34      result = dre_app_create_cgroup(h, cgroup);
35      if(result) goto error;
36
37      /* close handler */
38      if(h) {
39              result = dre_app_close(h);
40              if(result) goto error;
41      }
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_create_cgroup.c を参照。)

プログラムの引数でコンテナ名<name>を指定した場合、26 行目の dre_app_open()で、<name>のハンドルを取得します。34 行目の dre_app_create_cgroup()で、コントロールグループ<cgroup>を生成します。コントロールグループ<cgroup>は、実行時に<name>を指定した場合は<name>上に生成され、<name>を省略した場合はホスト OS 上に生成されます。<name>を指定した場合、39 行目の dre_app_close()で<name>のハンドルを解放します。

使用方法

dre_app_create_cgroup　-g <cgroup>　[-n <name>]

引数

<cgroup>	コントロールグループ
<name>	コンテナ名(オプション)

1.1.2.4　dre_app_create_container.c

概要

アプリケーションコンテナを作成するサンプル(“dre-app create -n <name>”に相当)

ソースコード解説

```
3   #include “dre_app.h”
4
5   int main(int argc, char **argv)
6   {
7           dre_return_t result;
8           char *name = NULL;
            ・・・(略)・・・
20          /* create container */
21          dre_app_data data = {DRE_APP_CONTAINER_TYPE_LXC, 2200, NULL};
22          result = dre_app_create_container(name, &data);
23          if(result) goto error;{
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_create_container.c を参照。)

21 行目で、作成するコンテナの設定値を指定します。上記の例では、以下の内容で設定を行っています。

- LXC コンテナを使用
- SSH ポート番号は 2200 を使用
- ベースイメージはデフォルトを使用

22 行目の dre_app_create_container()で、コンテナ名<name>(プログラムの引数で指定)と 15 行目で指定した設定値を使用して、コンテナ<name>の生成を行います。

使用方法

dre_app_create_container　-n <name>

引数

<name>　　コンテナ名

1.1.2.5　dre_app_destroy_cgroup.c

概要

コントロールグループを破棄するサンプル(“dre-app destroy [-n <name>] -g <cgroup>”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
10          char *cgroup = NULL;
            ・・・(略)・・・
24          /* open handler */
25          if(name) {
26                  h = dre_app_open(name);
27                  if(!h) {
28                          result = DRE_ERR_NULL_POINTER;
29                          goto error;
30                  }
31          }
32
33          /* destroy control group */
34          result = dre_app_destroy_cgroup(h, cgroup);
35          if(result) goto error;
36
37          /* close handler */
38          if(h) {
39                  result = dre_app_close(h);
40                  if(result) goto error;
```

```
41      }
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_destroy_cgroup.c を参照。)

プログラムの引数でコンテナ名<name>を指定した場合、26 行目の dre_app_open()で、<name>のハンドルを取得します。34 行目の dre_app_destroy_cgroup()で、コントロールグループ<cgroup>を破棄します。コントロールグループ<cgroup>は、実行時に<name>を指定した場合は<name>上のものが破棄され、<name>を省略した場合はホスト OS 上のものが破棄されます。<name>を指定した場合、39 行目の dre_app_close()で<name>のハンドルを解放します。

使用方法

dre_app_destroy_cgroup　-g <cgroup>　[-n <name>]

引数

<cgroup>	コントロールグループ
<name>	コンテナ名(オプション)

1.1.2.6　dre_app_destroy_container.c

概要

アプリケーションコンテナを破棄するサンプル(“dre-app destroy -n <name>”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          char *name = NULL;
            ・・・(略)・・・
20          /* destroy container */
21          result = dre_app_destroy_container(name);
22          if(result) goto error;{
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_destroy_container.c を参照。)

21 行目の dre_app_detroy_container()で、コンテナ<name>を破棄します。

使用方法

dre_app_destroy_container　-n <name>

引数

<name>	コンテナ名

1.1.2.7　dre_app_get_base_list.c

概要

存在するベースイメージを列挙するサンプル(“dre-app list --base”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_list_t list;
            ・・・(略)・・・
14          /* get base image list */
15          result = dre_app_get_base_list(&list);
16          if(result) goto error;
17          if(list) {
18                  while(*(list + i)) {
19                          printf(“%s¥n”, *(list + i));
20                          i++;
21                  }
22          }
23
24          /* release base image list */
25          if(list) {
26                  result = dre_app_release_base_list(list);
27                  if(result) goto error;
28          }
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_get_base_list.c を参照。)

15 行目の dre_app_get_base_list()で、システムに存在するベースイメージの名称を取得して、リスト化します。18-21 行目で、リスト化されたベースイメージの名称を列挙します。26 行目の dre_app_release_base_list()で、ベースイメージの名称のリストを解放します。

使用方法

dre_app_get_base_list

引数

なし

1.1.2.8　dre_app_get_container_list.c

概要

存在するアプリケーションコンテナを列挙するサンプル(“dre-app list”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
```

```
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_list_t list;
             ・・・(略)・・・
14          /* get container list */
15          result = dre_app_get_container_list(&list);
16          if(result) goto error;
17          if(list) {
18                  while(*(list + i)) {
19                          printf("%s¥n", *(list + i));
20                          i++;
21                  }
22          }
23
24          /* release container list */
25          if(list) {
26                  result = dre_app_release_container_list(list);
27                  if(result) goto error;
28          }
             ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/list.c を参照。)

15 行目の dre_app_get_container_list()で、システムに存在するコンテナの名称を取得して、リスト化します。18-21 行目で、リスト化されたコンテナの名称を列挙します。26 行目の dre_app_release_container_list()で、コンテナの名称のリストを解放します。

使用方法

dre_app_get_container_list

引数

なし

1.1.2.9　dre_app_get_registered_pid_list.c

概要

/proc/daware に登録されたプロセスの PID を列挙するサンプル(“dre-app pidlist -n <name>”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          dre_app_pid_list_t plist;
```

```
10      char *name = NULL;
        ・・・(略)・・・
22      /* open handler */
23      h = dre_app_open(name);
24      if(!h) {
25          result = DRE_ERR_NULL_POINTER;
26          goto error;
27      }
28
29      /* get pid list */
30      result = dre_app_get_registered_pid_list(h, &plist);
31      if(result) goto error;
32      if(plist) {
33          int j = 0;
34          printf("Registered process(es) on  '%s':¥n", name);
35          while(plist[j]) {
36              printf("- pid:%d(vpid:%d)¥n", (*plist[j]).pid, (*plist[j]).vpid);
37              j++;
38          }
39      } else {
40          printf("No process registered on  '%s':¥n", name);
41      }
42
43      /* release pid list */
44      if(list) {
45          result = dre_app_release_registered_pid_list(list);
46          if(result) goto error;
47      }
48
49      /* close handler */
50      result = dre_app_close(h);
51      if(result) goto error;
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_get_container_list.c を参照。)

23 行目の dre_app_open()で、コンテナ<name>のハンドルを取得します。30 行目の dre_app_get_registered_pid_list()で、/proc/daware に登録されたプロセスの PID を取得して、リスト化します。33-38 行目で、リスト化された PID を列挙します。45 行目の dre_app_release_registered_pid_list()で、PID のリストを解放します。50 行目の dre_app_close()で、<name>のハンドルを解放します。

使用方法

dre_app_get_registered_pid_list　-n <name>

引数

<name>	コンテナ名

1.1.2.10　dre_app_get_resource.c

概要

アプリケーションコンテナが使用するリソースの情報を取得するサンプル(“dre-app usage -n <name> -pid <pid> [-g <cgroup>]”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_app_resource_value_t resource;
 8          dre_return_t result:
 9          dre_app_handle *h = NULL;
10          char *name = NULL;
11          char *type = NULL;
            ・・・(略)・・・
37          /* open handler */
38          h = dre_app_open(name);
39          if(!h) {
40                  result = DRE_ERR_NULL_POINTER;
41                  goto error;
42          }
43
44          /* register process */
45          if(pid > 0) {
46                  result = dre_app_register_pid(h, pid);
47                  if(result) goto error;
48          }
49
50          /* get resource usage */
51          if(!strcmp(type, “cpu”)) {
52                  resource.type = DRE_APP_CONTAINER_RESOURCE_CPU;
53                  result = dre_app_get_resource(h, pid, &resource);
54                  if(result) goto error;
55                  printf(“cpu usage:       %d%%¥n”, resource.udata.cpu.percent);
56          } else if(!strcmp(type, “core”)) {
57                  resource.type = DRE_APP_CONTAINER_RESOURCE_CPU_CORE;
58                  result = dre_app_get_resource(h, pid, &resource);
59                  if(result) goto error;
60                  printf(“assined cpu core(s):       %s¥n”, resource.udata.cpu.cores);
61          } else if(!strcmp(type, “mem”)) {
62                  resource.type = DRE_APP_CONTAINER_RESOURCE_MEM;
63                  result = dre_app_get_resource(h, pid, &resource);
64                  if(result) goto error;
65                  printf(“memory usage: %dBytes¥n”, resource.udata.mem.usage_in_bytes);
66          } else if(!strcmp(type, “memsw”)) {
67                  resource.type = DRE_APP_CONTAINER_RESOURCE_MEMSW;
68                  result = dre_app_get_resource(h, pid, &resource);
69                  if(result) goto error;
```

```
70              printf("memory+swap usage: %dBytes¥n", resource.udata.mem.memsw_usage_in_bytes);
71      } else {
72              printf("Invalid resource type: %s¥n", type);
73              result = DRE_ERR_INVALID_ARGUMENT;
74              goto error;
75      }
76
77      /* unregister process */
78      if(pid > 0) {
79              result = dre_app_unregister_pid(h, pid);
80              if(result) goto error;
81      }
82
83      /* close handler */
84      result = dre_app_close(h);
85      if(result) goto error;
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_get_resource.c を参照。)

38 行目の dre_app_open()で、コンテナ<name>のハンドルを取得します。<pid>を指定した場合、46 行目の dre_app_register_pid()で、ホスト OS で割り当てられたプロセス ID(PID)を、アプリケーションコンテナ上で割り当てられたプロセス ID (VPID)と紐付けます。51-75 行目の dre_app_get_resource()で、リソース使用量を取得し、出力します。dre_app_get_resource()は、引数によって以下の動作をします。

- <cgroup>を指定した場合は、アプリケーションコンテナ<name>に存在するコントロールグループ<cgroup>が使用するリソースを指定します。
  - 上記に加えて<pid>を指定した場合は、当該<cgroup>にバインドされたプロセス<pid>が使用するリソースを指定します。
- <cgroup>を省略した場合は、<name>全体が使用するリソースを指定します。
  - 上記に加えて<pid>を指定した場合は、<name>上のプロセス<pid>が使用するリソースを指定します。

<pid>を指定した場合、79 行目の dre_app_unregister_pid()で PID と VPID の紐付けを解除します。84 行目の dre_app_close()で<name>のハンドルを解放します。

使用方法

dre_app_get_resource　-n <name>　<resource_type>　[--pid <pid>]　[-g <cgroup>]

引数

<name>	コンテナ名
<resource_type>	リソースの種類
	"cpu"　:　CPU 使用率
	"core"　:　CPU コア
	"mem"　:　メモリ使用量
	"memsw"　:　メモリとスワップ領域の使用量
<pid>	プロセス ID
<cgroup>	コントロールグループ(オプション)

1.1.2.11　dre_app_get_snapshot_list.c

概要

アプリケーションコンテナのスナップショットを列挙するサンプル(“dre-app list --snapshot -n <name>”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_app_handle *h = NULL;
 9          dre_list_t list;
10          char *name = NULL;
            ・・・(略)・・・
22          /* open handler */
23          h = dre_app_open(name);
24          if(!h) {
25                  result = DRE_ERR_NULL_POINTER;
26                  goto error;
27          }
28
29          /* get snapshot list */
30          result = dre_app_get_snapshot_list(h, &list);
31          if(result) goto error;
32          if(list) {
33                  int j = 0;
34                  while(*(list + j)) {
35                          printf(“%s¥n”, *(list + j));
36                          j++;
37                  }
38          }
39
40          /* release snapshot list */
41          if(list) {
42                  result = dre_app_release_snapshot_list(list);
43                  if(result) goto error;
44          }
45
46          /* close handler */
47          result = dre_app_close(h);
48          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_get_snapshot_list.c を参照。)

23 行目の dre_app_open()で、コンテナ<name>のハンドルを取得します。30 行目の dre_app_get_snapshot_list()で、システムに存在するスナップショットの名称を取得して、リスト化します。33-37 行目で、リスト化された

スナップショットの名称を列挙します。42 行目の dre_app_release_snapshot_list()で、スナップショットの名称のリストを解放します。47 行目の dre_app_close()で、<name>のハンドルを解放します。

使用方法

dre_app_get_snapshot_list　-n <name>

引数

<name>　　コンテナ名

1.1.2.12　dre_app_get_status

概要

アプリケーションコンテナの状態を取得するサンプル(“dre-app status -n <name>”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
             ・・・(略)・・・
21          /* open handler */
22          h = dre_app_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* get status */
29          result = dre_app_get_status(h);
30          if(result == DRE_APP_CONTAINER_STATUS_STOP) {
31                  printf(“’%s’ is STOPPED.¥n”, name);
32          } else if(result == DRE_APP_CONTAINER_STATUS_RUNNING) {
33                  printf(“’%s’ is RUNNING.¥n”, name);
34          } else if(result == DRE_APP_CONTAINER_STATUS_SUSPENDED) {
35                  printf(“’%s’ is SUSPENDED.¥n”, name);
36          } else if(result == DRE_APP_CONTAINER_STATUS_BUSY) {
37                  printf(“’%s’ is BUSY.¥n”, name);
38          } else {
39                  result = DRE_ERR_CONTAINER_UNKNOWN_STATUS;
40                  goto error;
41          }
42
43          /* close handler */
44          result = dre_app_close(h);
```

```
45      if(result) goto error;
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_get_status.c を参照。)

22 行目の dre_app_open()で、コンテナ<name>のハンドルを取得します。29 行目の dre_app_get_status()で、<name>の状態を取得し、30-41 行目で結果を出力します。44 行目の dre_app_close()で、<name>のハンドルを解放します。

使用方法

dre_app_get_status　-n <name>

引数

<name>　　コンテナ名

1.1.2.13　dre_app_load_snapshot.c

概要

アプリケーションコンテナをスナップショットの状態に復元するサンプル(“dre-app load -n <name> -t <tag>”に相当)

ソースコード解説

```
3   #include “dre_app.h”
4
5   int main(int argc, char **argv)
6   {
7       dre_return_t result:
8       dre_app_handle *h = NULL;
9       char *name = NULL;
10      char *snapshot = NULL;
        ・・・(略)・・・
24      /* open handler */
25      h = dre_app_open(name);
26      if(!h) {
27          result = DRE_ERR_NULL_POINTER;
28          goto error;
29      }
30
31      /* load snapshot */
32      result = dre_app_load_snapshot(h, snapshot);
33      if(result) goto error;
34
35      /* close handler */
36      result = dre_app_close(h);
37      if(result) goto error;
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_load_snapshot.c を参照。)

25 行目の dre_app_open()で、コンテナ<name>のハンドルを取得します。32 行目の dre_app_load_snapshot()で、<name>をスナップショット<tag>で復元します。36 行目の dre_app_close()で、<name>のハンドルを解放します。

使用方法

dre_app_load_snapshot　-n <name>　-t <tag>

引数

<name>	コンテナ名
<tag>	スナップショット名

1.1.2.14　dre_app_remove_base.c

概要

ベースイメージを削除するサンプル(“dre-app remove --base <base_image>”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *base = NULL;
            ・・・(略)・・・
21          /* remove base image */
22          result = dre_app_remove_base(base);
23          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_remove_base.c を参照。)

22 行目の dre_app_remove_base()で、ベースイメージ<base_image>を破棄します。

使用方法

dre_app_remove_base　-b <base_image>

引数

<base_image>	ベースイメージ名

1.1.2.15　dre_app_remove_snapshot.c

概要

アプリケーションコンテナのスナップショットを削除するサンプル(“dre-app remove --snapshot -n <name> -t <tag>”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7        dre_return_t result:
 8        dre_app_handle *h = NULL;
 9        char *name = NULL;
10        char *snapshot = NULL;
          ・・・(略)・・・
24        /* open handler */
25        h = dre_app_open(name);
26        if(!h) {
27              result = DRE_ERR_NULL_POINTER;
28              goto error;
29        }
30
31        /* remove snapshot */
32        result = dre_app_remove_snapshot(h, snapshot);
33        if(result) goto error;
34
35        /* close handler */
36        result = dre_app_close(h);
37        if(result) goto error;
          ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_remove_snapshot.c を参照。)

25 行目の dre_app_open()で、コンテナ<name>のハンドルを取得します。32 行目の dre_app_remove_snapshot()で、<name>のスナップショット<tag>を削除します。36 行目の dre_app_close()で、<name>のハンドルを解放します。

使用方法

dre_app_remove_snapshot　-n <name>　-t <tag>

引数

<name>	コンテナ名
<tag>	スナップショット名

1.1.2.16　dre_app_resume.c

概要

一時停止中のアプリケーションコンテナの動作を再開するサンプル(“dre-app resume -n <name>”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
```

```
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
            ・・・(略)・・・
21          /* open handler */
22          h = dre_app_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* resume container */
29          result = dre_app_resume(h);
30          if(result) goto error;
31
32          /* close handler */
33          result = dre_app_close(h);
34          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_resume.c を参照。)

22 行目の dre_app_open()で、コンテナ<name>のハンドルを取得します。29 行目の dre_app_resume()で、<name>の動作を再開します。33 行目の dre_app_close()で、<name>のハンドルを解放します。

使用方法

dre_app_resume　-n <name>

引数

<name>　　コンテナ名

1.1.2.17　dre_app_save_snapshot.c

概要

アプリケーションコンテナのスナップショットを作成するサンプル(“dre-app save -n <name> -t <tag>”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
10          char *snapshot = NULL;
```

```
        ・・・(略)・・・
24      /* open handler */
25      h = dre_app_open(name);
26      if(!h) {
27              result = DRE_ERR_NULL_POINTER;
28              goto error;
29      }
30
31      /* save snapshot */
32      result = dre_app_save_snapshot(h, snapshot);
33      if(result) goto error;
34
35      /* close handler */
36      result = dre_app_close(h);
37      if(result) goto error;
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_save_snapshot.c を参照。)

25 行目の dre_app_open()で、コンテナ<name>のハンドルを取得します。32 行目の dre_app_save_snapshot()で、<name>のスナップショット<tag>を作成します。36 行目の dre_app_close()で、<name>のハンドルを解放します。

使用方法

dre_app_save_snapshot　-n <name>　-t <tag>

引数

<name>	コンテナ名
<tag>	スナップショット名

1.1.2.18　dre_app_set_resource.c

概要

アプリケーションコンテナが使用するリソースに制限をかけるサンプル(“dre-app limit <resource_type> <limitation> -n <name> [-g <cgroup>]”に相当)

ソースコード解説

```
3   #include “dre_app.h”
4
5   int main(int argc, char **argv)
6   {
7           dre_return_t result:
8           dre_app_handle *h = NULL;
9           dre_app_resource_value_t resource;
10          char *name = NULL;
11          char *cgroup = NULL;
12          char *type = NULL;
13          char *limitation = NULL;
```

```
      ・・・(略)・・・
39    /* set resource usage */
40    if(!strcmp(type, "cpu")) {
41          resource.type = DRE_APP_CONTAINER_RESOURCE_CPU;
42          resource.udata.cpu.percent = atoi(limitation);
43    } else if(!strcmp(type, "core")) {
44          resource.type = DRE_APP_CONTAINER_RESOURCE_CPU_CORE;
45          resource.udata.cpu.cores = limitation;
46    } else if(!strcmp(type, "mem")) {
47          resource.type = DRE_APP_CONTAINER_RESOURCE_MEM;
48          resource.udata.mem.usage_in_bytes = atoi(limitation);
49    } else if(!strcmp(type, "memsw")) {
50          resource.type = DRE_APP_CONTAINER_RESOURCE_MEMSW;
51          resource.udata.mem.memsw_usage_in_bytes = atoi(limitation);
52    } else {
53          printf("Invalid resource type: %s¥n", type);
54          result = DRE_ERR_INVALID_ARGUMENT;
55          goto error;
56    }
57
58    /* open handler */
59    if(name) {
60          h = dre_app_open(name);
61          if(!h) {
62                result = DRE_ERR_NULL_POINTER;
63                goto error;
64          }
65    }
66
67    /* set limitation */
68    result = dre_app_set_resource(h, cgroup, resource);
69    if(result) goto error;
70    printf("Set limitation of %s.¥n", type);
71
72    /* close handler */
73    if(h) {
74          result = dre_app_close(h);
75          if(result) goto error;
76    }
      ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_set_resource.c を参照。)

40-56 行目で、制限を掛けるリソースと使用量を指定します。これは、実行時の引数によって以下の動作をします。

- 〈cgroup〉を指定した場合は、アプリケーションコンテナ〈name〉に存在するコントロールグループ〈cgroup〉が使用するリソースを指定します。
- 〈cgroup〉を省略した場合は、〈name〉全体が使用するリソースを指定します。

60 行目の dre_app_open()で、コンテナ<name>のハンドルを取得します。68 行目の dre_app_set_resource()で、40-56 行目で指定した内容でリソース使用量の制限を行います。
74 行目の dre_app_close()で、<name>のハンドルを解放します。

使用方法

dre_app_set_resource　-n <name> <resource_type> <limitation> [-g <cgroup>]

引数

<name>	コンテナ名
<resource_type>	リソースの種類
	“cpu” : CPU 使用率
	“core” : CPU コア
	“mem” : メモリ使用量
	“memsw” : メモリとスワップ領域の使用量
<limitation>	リソース使用量の制限
<cgroup>	コントロールグループ(オプション)

1.1.2.19 dre_app_start.c

概要

アプリケーションコンテナの動作を開始するサンプル(“dre-app start -n <name>”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
            ・・・(略)・・・
21          /* open handler */
22          h = dre_app_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* start container */
29          result = dre_app_start(h);
30          if(result) goto error;
31
32          /* close handler */
33          result = dre_app_close(h);
34          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_start.c を参照。)

22 行目の dre_app_open()で、コンテナ<name>のハンドルを取得します。29 行目の dre_app_start()で、<name>の動作を開始します。33 行目の dre_app_close()で、<name>のハンドルを解放します。

使用方法

dre_app_start　-n <name>

引数

<name>　コンテナ名

1.1.2.20　dre_app_stop.c

概要

アプリケーションコンテナの動作を停止するサンプル(“dre-app stop -n <name>”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_app_handle *h;
            ・・・(略)・・・
21          /* open handler */
22          h = dre_app_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* stop container */
29          result = dre_app_stop(h);
30          if(result) goto error;
31
32          /* close handler */
33          result = dre_app_close(h);
34          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_stop.c を参照。)

22 行目の dre_app_open()で、コンテナ<name>のハンドルを取得します。29 行目の dre_app_stop()で、<name>の動作を停止します。33 行目の dre_app_close()で、<name>のハンドルを解放します。

使用方法

dre_app_stop　-n <name>

引数

〈name〉　　コンテナ名

1.1.2.21　dre_app_suspend.c

概要

動作中のアプリケーションコンテナを一時停止するサンプル(“dre-app suspend -n 〈name〉”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          char *name = NULL;
            ・・・(略)・・・
21          /* open handler */
22          h = dre_app_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* suspend container */
29          result = dre_app_suspend(h);
30          if(result) goto error;
31
32          /* close handler */
33          result = dre_app_close(h);
34          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_suspend.c を参照。)

22 行目の dre_app_open()で、コンテナ〈name〉のハンドルを取得します。29 行目の dre_app_suspend()で、〈name〉の動作を一時停止します。33 行目の dre_app_close()で、〈name〉のハンドルを解放します。

使用方法

dre_app_suspend　-n 〈name〉

引数

〈name〉　　コンテナ名

1.1.2.22　dre_app_unbind_pid.c

概要

コントロールグループからプロセスをアンバインドするサンプル(“dre-app unbind [-n <name>] -g <cgroup> -pid <pid>”に相当)

ソースコード解説

```
 3  #include “dre_app.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7        dre_return_t result:
 8        dre_app_handle *h = NULL;
 9        char *name = NULL;
10        char *cgroup = NULL;
          ・・・(略)・・・
27        /* open handler */
28        if(name) {
29              h = dre_app_open(name);
30              if(!h) {
31                    result = DRE_ERR_NULL_POINTER;
32                    goto error;
33              }
34        }
35
36        /* register pid */
37        result = dre_app_register_pid(h, pid);
38        if(result) goto error;
39
40        /* unbind from control group */
41        result = dre_app_unbind_pid(h, cgroup, pid);
42        if(result) goto error;
43
44        /* unregister pid */
45        result = dre_app_unregister_pid(h, pid);
46        if(result) goto error;
47
48        /* close handler */
49        if(h) {
50              result = dre_app_close(h);
51              if(result) goto error;
52        }
          ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_unbind_pid.c を参照。)

プログラムの引数でコンテナ名<name>を指定した場合、29 行目の dre_app_open()で、<name>のハンドルを取得します。37 行目の dre_app_register_pid()で、ホスト OS で割り当てられたプロセス ID(PID)を、アプリケーションコンテナ上で割り当てられたプロセス ID (VPID)と紐付けます。41 行目の dre_app_unbind_pid()で、<pid>をコントロールグループ<cgroup>からアンバインドします。この時、<cgroup>は、<name>が指定された場合は<name>上のものが使用され、<name>を省略した場合はホスト OS 上のものが使用されます。45 行目の dre_app_unregister_pid()で、PID と VPID の紐付けを解除します。<name>を指定した場合、50 行目の dre_app_close()で<name>のハンドルを解放します。

使用方法

dre_app_unbind_pid　-g <cgroup>　--pid <pid>　[-n <name>]

引数

<cgroup>	コントロールグループ名
<pid>	プロセス ID
<name>	コンテナ名(オプション)

1.1.2.23　dre_app_unset_resource.c

概要

アプリケーションコンテナのリソース制限を解除するサンプル("dre-app unlimit <resource_type> -n <name> [-g <cgroup>]"に相当)

ソースコード解説

```
 3  #include "dre_app.h"
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_app_resource_value_t resource;
 8          dre_return_t result:
 9          dre_app_handle *h = NULL;
10          char *name = NULL;
11          char *cgroup = NULL;
12          char *type = NULL;
            ・・・(略)・・・
34          /* unset resource usage */
35          if(!strcmp(type, "cpu")) {
36                  resource.type = DRE_APP_CONTAINER_RESOURCE_CPU;
37          } else if(!strcmp(type, "core")) {
38                  resource.type = DRE_APP_CONTAINER_RESOURCE_CPU_CORE;
39          } else if(!strcmp(type, "mem")) {
40                  resource.type = DRE_APP_CONTAINER_RESOURCE_MEM;
41          } else if(!strcmp(type, "memsw")) {
42                  resource.type = DRE_APP_CONTAINER_RESOURCE_MEMSW;
43          } else {
44                  printf("Invalid resource type: %s¥n", type);
45                  result = DRE_ERR_INVALID_ARGUMENT;
46                  goto error;
47          }
48
49          /* open handler */
50          if(name) {
51                  h = dre_app_open(name);
52                  if(!h) {
53                          result = DRE_ERR_NULL_POINTER;
54                          goto error;
```

```
55              }
56      }
57
58      /* unset limitation */
59      result = dre_app_unset_resource(h, cgroup, resource);
60      if(result) goto error;
61      printf("Unset limitation of %s.¥n", type);
62
63      /* close handler */
64      if(h) {
65              result = dre_app_close(h);
66              if(result) goto error;
67      }
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/app/dre_app_unset_resource.c を参照。)

35-47 行目で、制限を解除するリソースを指定します。これは、実行時の引数によって以下の動作をします。

- <cgroup>を指定した場合は、アプリケーションコンテナ<name>に存在するコントロールグループ<cgroup>が使用するリソースを指定します。
- <cgroup>を省略した場合は、<name>全体が使用するリソースを指定します。

51 行目の dre_app_open()で、コンテナ<name>のハンドルを取得します。59 行目の dre_app_unset_resource()で、35-47 行目で指定した内容でリソース使用量の制限を解除します。
65 行目の dre_app_close()で、<name>のハンドルを解放します。

使用方法

dre_app_unset_resource　-n <name> <resource_type> [-g <cgroup>]

引数

<name>	コンテナ名
<resource_type>	リソースの種類
	“cpu” : CPU 使用率
	“core” : CPU コア
	“mem” : メモリ使用量
	“memsw” : メモリとスワップ領域の使用量
<cgroup>	コントロールグループ(オプション)

1.2 システムコンテナの API をコールするサンプルプログラム

1.2.1サンプルプログラムの使用方法

libdre0-dev をインストールすると、以下のパスに本 API を使用したサンプルプログラムのソースコードがインストールされています。

/usr/share/doc/libdre0-dev/sample/sys

上記のパスで make コマンドを実行すると、サンプルプログラムがコンパイルできます。

1.2.2サンプルプログラムの解説

コンテナ API には、以下のサンプルプログラムが付属します。

dre_sys_bind_pid.c
システムコンテナ上のコントロールグループにプロセスをバインドするサンプル

dre_sys_checkpoint.c
システムコンテナのチェックポイントを作成するサンプル

dre_sys_commit_snapshot.c
システムコンテナのスナップショットを使用して、ベースイメージを新規作成するサンプル

dre_sys_create_cgroup.c
システムコンテナ上にコントロールグループを作成するサンプル

dre_sys_create_container.c
システムコンテナを作成するサンプル

dre_sys_destroy_cgroup.c
システムコンテナ上のコントロールグループを破棄するサンプル

dre_sys_destroy_container.c
システムコンテナを破棄するサンプル

dre_sys_get_base_list.c
存在するベースイメージを列挙するサンプル

dre_sys_get_checkpoint_list.c
システムコンテナのチェックポイントを列挙するサンプル

dre_sys_get_container_list.c
存在するシステムコンテナを列挙するサンプル

dre_sys_get_resource.c
システムコンテナが使用するリソースの情報を取得するサンプル

dre_sys_get_snapshot_list.c
システムコンテナのスナップショットを列挙するサンプル

dre_sys_get_status.c
システムコンテナの状態を取得するサンプル

dre_sys_immigrate.c
システムコンテナのマイグレーション(マイグレーション先)を開始するサンプル

dre_sys_load_snapshot.c
システムコンテナをスナップショットの状態に復元するサンプル

dre_sys_migrate.c
システムコンテナのマイグレーションを開始するサンプル

dre_sys_remove_base.c

ベースイメージを削除するサンプル

dre_sys_remove_checkpoint.c

システムコンテナのチェックポイントを削除するサンプル

dre_sys_remove_snapshot.c

システムコンテナのスナップショットを削除するサンプル

dre_sys_restart.c

システムコンテナの動作をチェックポイントから再開するサンプル

dre_sys_resume.c

一時停止中のシステムコンテナの動作を再開するサンプル

dre_sys_save_snapshot.c

システムコンテナのスナップショットを作成するサンプル

dre_sys_set_resource.c

システムコンテナが使用するリソースに制限をかけるサンプル

dre_sys_share.c

システムコンテナを NFS サーバと共有するサンプル

dre_sys_start.c

システムコンテナの動作を開始するサンプル

dre_sys_stop.c

システムコンテナの動作を停止するサンプル

dre_sys_suspend.c

動作中のシステムコンテナを一時停止するサンプル

dre_sys_unbind_pid.c

システムコンテナ上のコントロールグループからプロセスをアンバインドするサンプル

dre_sys_unset_resource.c

システムコンテナのリソース制限を解除するサンプル

dre_sys_unshare.c

ローカルと NFS サーバにおけるシステムコンテナの共有を解除するサンプル

1.2.2.1 dre_sys_bind_pid.c

概要

コントロールグループにプロセスをバインドするサンプル(“dre-sys bind [-n <name>] -g <cgroup> -pid <pid>” に相当)

ソースコード解説

```
 3 #include “dre_sys.h”
 4
 5 int main(int argc, char **argv)
 6 {
 7         dre_return_t result;
 8         dre_sys_handle *h = NULL;
 9         char *name = NULL;
10         char *cgroup = NULL;
           ・・・(略)・・・
27         /* open handler */
28         if(name) {
```

```
29              h = dre_sys_open(name);
30              if(!h) {
31                      result = DRE_ERR_NULL_POINTER;
32                      goto error;
33              }
34      }
35
36      /* bind to control group */
37      result = dre_sys_bind_pid(h, cgroup, pid);
38      if(result) goto error;
39
40      /* close handler */
41      if(h) {
42              result = dre_sys_close(h);
43              if(result) goto error;
44      }
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_bind_pid.c を参照。)

プログラムの引数でコンテナ名<name>を指定した場合、29 行目の dre_sys_open()で、<name>のハンドルを取得します。37 行目の dre_sys_bind_pid()で、<pid>をコントロールグループ<cgroup>にバインドします。この時、<cgroup>は、<name>が指定された場合は<name>上のものが使用され、<name>を省略した場合はホスト OS 上のものが使用されます。<name>を指定した場合、42 行目の dre_sys_close()で<name>のハンドルを解放します。

使用方法

dre_sys_bind_pid　-g <cgroup>　--pid <pid>　[-n <name>]

引数

<cgroup>	コントロールグループ名
<pid>	プロセス ID
<name>	コンテナ名(オプション)

1.2.2.2　dre_sys_checkpoint.c

概要

システムコンテナのチェックポイントを作成するサンプル(“dre-sys checkpoint -n <name> -t <tag>”に相当)

ソースコード解説

```
3   #include “dre_sys.h”
4
5   int main(int argc, char **argv)
6   {
7           dre_return_t result:
8           dre_sys_handle *h = NULL;
9           char *name = NULL;
10          char *tag = NULL;
```

```
        ・・・(略)・・・
24      /* open handler */
25      h = dre_sys_open(name);
26      if(!h) {
27              result = DRE_ERR_NULL_POINTER;
28              goto error;
29      }
30
31      /* save checkpoint */
32      result = dre_sys_checkpoint(h, tag);
33      if(result) goto error;
34
35      /* close handler */
36      result = dre_sys_close(h);
37      if(result) goto error;
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_checkpoint.c を参照。)

25 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。32 行目の dre_sys_checkpoint()で、<name>のチェックポイント<tag>を作成します。36 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_checkpoint　-n <name>　-t <tag>

引数

<name>	コンテナ名
<tag>	チェックポイント名

1.2.2.3　dre_sys_commit_snapshot.c

概要

システムコンテナのスナップショットを使用して、ベースイメージを新規作成するサンプル(“dre-sys commit -n <name> -t <tag> -b <base_image>”に相当)

ソースコード解説

```
3   #include “dre_sys.h”
4
5   int main(int argc, char **argv)
6   {
7       dre_return_t result:
8       dre_sys_handle *h = NULL;
9       char *name = NULL;
10      char *snapshot = NULL;
11      char *base = NULL;
        ・・・(略)・・・
27      /* open handler */
```

```
28        h = dre_sys_open(name);
29        if(!h) {
30              result = DRE_ERR_NULL_POINTER;
31              goto error;
32        }
33
34        /* commit snapshot */
35        result = dre_sys_commit_snapshot(h, snapshot, base);
36        if(result) goto error;
37
38        /* close handler */
39        result = dre_sys_close(h);
40        if(result) goto error;
          ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_commit.c を参照。)

28 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。35 行目の dre_sys_commit_snapshot()で、<name>のスナップショット<tag>を使用してベースイメージ<base_image>を作成します。39 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_commit_snapshot　-n <name>　-t <tag>　-b <base_image>

引数

<name>	コンテナ名
<tag>	スナップショット名
<base_image>	作成するイメージ名

1.2.2.4　dre_sys_create_cgroup.c

概要

システムコンテナ上にコントロールグループを作成するサンプル(“dre-sys create -n <name> -g <cgroup>”に相当)

ソースコード解説

```
3   #include “dre_sys.h”
4
5   int main(int argc, char **argv)
6   {
7         dre_return_t result:
8         dre_sys_handle *h = NULL;
9         char *name = NULL;
10        char *cgroup = NULL;
          ・・・(略)・・・
24        /* open handler */
25        if(name) {
26              h = dre_sys_open(name);
```

```
27              if(!h) {
28                      result = DRE_ERR_NULL_POINTER;
29                      goto error;
30              }
31      }
32
33      /* create control group */
34      result = dre_sys_create_cgroup(h, cgroup);
35      if(result) goto error;
36
37      /* close handler */
38      if(h) {
39              result = dre_sys_close(h);
40              if(result) goto error;
41      }
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_create_cgroup.c を参照。)

プログラムの引数でコンテナ名<name>を指定した場合、26 行目の dre_sys_open()で、<name>のハンドルを取得します。34 行目の dre_sys_create_cgroup()で、コントロールグループ<cgroup>を生成します。コントロールグループ<cgroup>は、実行時に<name>を指定した場合は<name>上に生成され、<name>を省略した場合はホスト OS 上に生成されます。<name>を指定した場合、39 行目の dre_sys_close()で<name>のハンドルを解放します。

使用方法

dre_sys_create_cgroup　-g <cgroup>　[-n <name>]

引数

<cgroup>	コントロールグループ
<name>	コンテナ名(オプション)

1.2.2.5　dre_sys_create_container.c

概要

システムコンテナを作成するサンプル(“dre-sys create -n <name>”に相当)

ソースコード解説

```
3   #include “dre_sys.h”
4
5   int main(int argc, char **argv)
6   {
7           dre_return_t result;
8           char *name = NULL;
            ・・・(略)・・・
20          /* create container */
21          dre_sys_data data = {DRE_SYS_CONTAINER_TYPE_KVM, 512, NULL, NULL};
22          result = dre_sys_create_container(name, &data);
```

```
 23        if(result) goto error;
           ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_create_container.c を参照。)

21 行目で、作成するコンテナの設定値を指定します。上記の例では、以下の内容で設定を行っています。

- KVM を使用
- メモリは 512MB 使用
- ベースイメージはデフォルトを使用
- KVM イメージにインストールされた OS を使用する(別パーティションの OS は使用しない)

22 行目の dre_sys_create_container()で、コンテナ名<name>(プログラムの引数で指定)と 15 行目で指定した設定値を使用して、コンテナ<name>の生成を行います。

使用方法

der_sys_create_container　-n <name>

引数

<name>	コンテナ名

1.2.2.6　dre_sys_destroy_cgroup.c

概要

システムコンテナ上のコントロールグループを破棄するサンプル("dre-sys destroy -n <name> -g <cgroup>"に相当)

ソースコード解説

```
  3  #include "dre_sys.h"
  4
  5  int main(int argc, char **argv)
  6  {
  7        dre_return_t result;
  8        dre_sys_handle *h = NULL;
  9        char *name = NULL;
 10        char *cgroup = NULL;
           ・・・(略)・・・
 24        /* open handler */
 25        if(name) {
 26              h = dre_sys_open(name);
 27              if(!h) {
 28                    result = DRE_ERR_NULL_POINTER;
 29                    goto error;
 30              }
 31        }
 32
 33        /* destroy control group */
 34        result = dre_sys_destroy_cgroup(h, cgroup);
 35        if(result) goto error;
```

```
 36
 37        /* close handler */
 38        if(h) {
 39              result = dre_sys_close(h);
 40              if(result) goto error;
 41        }
           ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_destroy_cgoup.c を参照。)

プログラムの引数でコンテナ名<name>を指定した場合、26 行目の dre_sys_open()で、<name>のハンドルを取得します。34 行目の dre_sys_destroy_cgroup()で、コントロールグループ<cgroup>を破棄します。コントロールグループ<cgroup>は、実行時に<name>を指定した場合は<name>上のものが破棄され、<name>を省略した場合はホスト OS 上のものが破棄されます。<name>を指定した場合、39 行目の dre_sys_close()で<name>のハンドルを解放します。

使用方法

dre_sys_destroy_container　-g <cgroup>　[-n <name>]

引数

<cgroup>	コントロールグループ
<name>	コンテナ名(オプション)

1.2.2.7　dre_sys_destroy_container.c

概要

システムコンテナを破棄するサンプル(“dre-sys destroy -n <name>”に相当)

ソースコード解説

```
  3  #include “dre_sys.h”
  4
  5  int main(int argc, char **argv)
  6  {
  7        dre_return_t result;
  8        char *name = NULL;
           ・・・(略)・・・
 20        /* destroy container */
 21        result = dre_sys_destroy_container(name);
 22        if(result) goto error;{
           ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_destroy_container.c を参照。)

21 行目の dre_sys_detroy_container()で、コンテナ<name>を破棄します。

使用方法

dre_sys_destroy_container　-n <name>

引数

〈name〉　コンテナ名

1.2.2.8 dre_sys_get_base_list.c

概要

存在するベースイメージを列挙するサンプル(“dre-sys list --base“に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_list_t list;
            ・・・(略)・・・
14          /* get base image list */
15          result = dre_sys_get_base_list(&list);
16          if(result) goto error;
17          if(list) {
18                  while(*(list + i)) {
19                          printf(“%s¥n”, *(list + i));
20                          i++;
21                  }
22          }
23
24          /* release base image list */
25          if(list) {
26                  result = dre_sys_release_base_list(list);
27                  if(result) goto error;
28          }
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_get_base_list.c を参照。)

15 行目の dre_sys_get_base_list()で、システムに存在するベースイメージの名称を取得して、リスト化します。18-21 行目で、リスト化されたベースイメージの名称を列挙します。26 行目の dre_sys_release_base_list()で、ベースイメージの名称のリストを解放します。

使用方法

dre_sys_get_base_list

引数

なし

1.2.2.9 dre_sys_get_checkpoint_list.c

概要

システムコンテナのチェックポイントを列挙するサンプル(“dre-sys list --checkpoint“に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_app_handle *h = NULL;
 9          dre_list_t list;
10          char *name = NULL;
            ・・・(略)・・・
22          /* open handler */
23          h = dre_sys_open(name);
24          if(!h) {
25                  result = DRE_ERR_NULL_POINTER;
26                  goto error;
27          }
28
29          /* get checkpoint list */
30          result = dre_sys_get_checkpoint_list(h, &list);
31          if(result) goto error;
32          if(list) {
33                  int j = 0;
34                  while(*(list + j)) {
35                          printf(“%s¥n”, *(list + j));
36                          j++;
37                  }
38          }
39
40          /* release checkpoint list */
41          if(list) {
42                  result = dre_sys_release_checkpoint_list(list);
43                  if(result) goto error;
44          }
45
46          /* close handler */
47          result = dre_sys_close(h);
48          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_get_checkpoint_list.c を参照。)

23 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。30 行目の dre_sys_get_checkpoint_list()で、システムに存在するチェックポイントの名称を取得して、リスト化します。33-37 行目で、リスト化されたチェックポイントの名称を列挙します。42 行目の dre_sys_release_checkpoint_list()で、チェックポイントの名称のリストを解放します。47 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_get_checkpoint_list -n <name>

引数

<name>　　コンテナ名

1.2.2.10 dre_sys_get_container_list.c

概要

存在するシステムコンテナを列挙するサンプル(“dre-sys list“に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_list_t list;
            ・・・(略)・・・
14          /* get container list */
15          result = dre_sys_get_container_list(&list);
16          if(result) goto error;
17          if(list) {
18                  while(*(list + i)) {
19                          printf(”%s¥n”, *(list + i));
20                          i++;
21                  }
22          }
23
24          /* release container list */
25          if(list) {
26                  result = dre_sys_release_container_list(list);
27                  if(result) goto error;
28          }
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_get_container_list.c を参照。)

15 行目の dre_sys_get_container_list()で、システムに存在するコンテナの名称を取得して、リスト化します。18-21 行目で、リスト化されたコンテナの名称を列挙します。26 行目の dre_sys_release_container_list()で、コンテナの名称のリストを解放します。

使用方法

dre_sys_get_container_list

引数

なし

1.2.2.11 dre_sys_get_resource.c

概要

システムコンテナが使用するリソースの情報を取得するサンプル(“dre-sys usage -n <name> -pid <pid> [-a <application_container>]”に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_sys_resource_value_t resource;
 8          dre_return_t result:
 9          dre_sys_handle *h = NULL;
10          char *name = NULL;
11          char *type = NULL;
12          char *cgroup = NULL;
13          char *app = NULL;
            ・・・(略)・・・
40          /* open handler */
41          h = dre_sys_open(name);
42          if(!h) {
43                  result = DRE_ERR_NULL_POINTER;
44                  goto error;
45          }
46
47          /* get resource usage */
48          if(!strcmp(type, “cpu”)) {
49                  resource.type = DRE_SYS_CONTAINER_RESOURCE_CPU;
50                  result = dre_sys_get_resource(h, pid, &resource);
51                  if(result) goto error;
52                  printf(“cpu usage:       %d%%¥n”, resource.udata.cpu.percent);
53          } else if(!strcmp(type, “core”)) {
54                  resource.type = DRE_SYS_CONTAINER_RESOURCE_CPU_CORE;
55                  result = dre_sys_get_resource(h, pid, &resource);
56                  if(result) goto error;
57                  printf(“assined cpu core(s):       %s¥n”, resource.udata.cpu.cores);
58          } else if(!strcmp(type, “mem”)) {
59                  resource.type = DRE_SYS_CONTAINER_RESOURCE_MEM;
60                  result = dre_sys_get_resource(h, pid, &resource);
61                  if(result) goto error;
62                  printf(“memory usage: %dBytes¥n”, resource.udata.mem.usage_in_bytes);
63          } else if(!strcmp(type, “memsw”)) {
64                  resource.type = DRE_SYS_CONTAINER_RESOURCE_MEMSW;
65                  result = dre_sys_get_resource(h, pid, &resource);
66                  if(result) goto error;
67                  printf(“memory+swap usage: %dBytes¥n”, resource.udata.mem.memsw_usage_in_bytes);
68          } else {
```

```
69              printf("Invalid resource type: %s¥n", type);
70              result = DRE_ERR_INVALID_ARGUMENT;
71              goto error;
72          }
73
74          /* close handler */
75          result = dre_sys_close(h);
76          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_get_resource.c を参照。)

41 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。48-72 行目の dre_sys_get_resource()で、リソース使用量を取得し、出力します。dre_sys_get_resource()は、引数によって以下の動作をします。

- <application_container>と<cgroup>を指定した場合は、システムコンテナ<name>上のアプリケーションコンテナ<application_container>に存在するコントロールグループ<cgroup>が使用するリソースを指定します。
  - 上記に加えて<pid>を指定した場合は、当該<cgroup>にバインドされたプロセス<pid>が使用するリソースを指定します。
- <application_container>を省略し、<cgroup>を指定した場合は、<name>に存在するコントロールグループ<cgroup>が使用するリソースを指定します。
  - 上記に加えて<pid>を指定した場合は、当該<cgroup>にバインドされたプロセス<pid>が使用するリソースを指定します。
- <application_container>を指定し、<cgroup>を省略した場合は、<application_container>全体が使用するリソースを指定します。
  - 上記に加えて<pid>を指定した場合は、<application_container>上のプロセス<pid>が使用するリソースを指定します。
- <application_container>と<cgroup>を省略した場合は、<name>全体のリソースを指定します。
  - 上記に加えて<pid>を指定した場合は、<name>上のプロセス<pid>が使用するリソースを指定します。

75 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

```
dre_sys_get_resource   -n <name> <resource_type> [--pid <pid>] [-g <cgroup>]
                       [-a <application_container>]
```

引数

<name>	コンテナ名
<resource_type>	リソースの種類 "cpu"　:　CPU 使用率 "core"　:　CPU コア "mem"　:　メモリ使用量 "memsw"　:　メモリとスワップ領域の使用量
<pid>	プロセス ID(オプション)
<cgroup>	コントロールグループ(オプション)
<application_container>	アプリケーションコンテナ名(オプション)

1.2.2.12　dre_sys_get_snapshot_list.c

概要

システムコンテナのスナップショットを列挙するサンプル(“dre-sys list --snapshot -n <name>”に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          dre_list_t list;
10          char *name = NULL;
            ・・・(略)・・・
22          /* open handler */
23          h = dre_sys_open(name);
24          if(!h) {
25                  result = DRE_ERR_NULL_POINTER;
26                  goto error;
27          }
28
29          /* get snapshot list */
30          result = dre_sys_get_snapshot_list(h, &list);
31          if(result) goto error;
32          if(list) {
33                  int j = 0;
34                  while(*(list + j)) {
35                          printf(“%s¥n”, *(list + j));
36                          j++;
37                  }
38          }
39
40          /* release snapshot list */
41          if(list) {
42                  result = dre_sys_release_snapshot_list(list);
43                  if(result) goto error;
44          }
45
46          /* close handler */
47          result = dre_sys_close(h);
48          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_get_snapshot_list.c を参照。)

23 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。30 行目の dre_sys_get_snapshot_list()で、システムに存在するスナップショットの名称を取得して、リスト化します。33-37 行目で、リスト化された

スナップショットの名称を列挙します。42 行目の dre_sys_release_snapshot_list()で、スナップショットの名称のリストを解放します。47 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_get_snapshot_list　-n <name>

引数

<name>	コンテナ名

1.2.2.13　dre_sys_get_status.c

概要

システムコンテナの状態を取得するサンプル(“dre-sys status -n <name>”に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7      dre_return_t result;
 8      dre_sys_handle *h = NULL;
 9      char *name = NULL;
        ・・・(略)・・・
21      /* open handler */
22      h = dre_sys_open(name);
23      if(!h) {
24          result = DRE_ERR_NULL_POINTER;
25          goto error;
26      }
27
28      /* get status */
29      result = dre_sys_get_status(h);
30      if(result == DRE_SYS_CONTAINER_STATUS_STOP) {
31          printf(“’%s’ is STOPPED.¥n”, name);
32      } else if(result == DRE_SYS_CONTAINER_STATUS_RUNNING) {
33          printf(“’%s’ is RUNNING.¥n”, name);
34      } else if(result == DRE_SYS_CONTAINER_STATUS_SUSPENDED) {
35          printf(“’%s’ is SUSPENDED.¥n”, name);
36      } else if(result == DRE_SYS_CONTAINER_STATUS_BUSY) {
37          printf(“’%s’ is BUSY.¥n”, name);
38      } else {
39          result = DRE_ERR_CONTAINER_UNKNOWN_STATUS;
40          goto error;
41      }
42
43      /* close handler */
44      result = dre_sys_close(h);
```

```
45      if(result) goto error;
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_get_status.c を参照。)

22 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。29 行目の dre_sys_get_status()で、<name>の状態を取得し、30-41 行目で結果を出力します。44 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_get_status　-n <name>

引数

<name>	コンテナ名

1.2.2.14　dre_sys_immigration_start.c

概要

システムコンテナのマイグレーション(マイグレーション先)を開始するサンプル("dre-sys immigrate -n <name> -p <port>"に相当)

ソースコード解説

```
3   #include "dre_sys.h"
4
5   int main(int argc, char **argv)
6   {
7       dre_return_t result;
8       int port;
        ・・・(略)・・・
23      /* start immigration */
24      result = dre_sys_immigration_start(h, port);
25      if(result) goto error;
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_immigration_start.c を参照。)

24 行目の dre_sys_immigration_start()で、<name>のマイグレーションに使用する VM の動作を開始します。

使用方法

dre_sys_immigrate　-n <name>　-p <port>

引数

<name>	コンテナ名
<port>	マイグレーションに使用するポート番号

1.2.2.15　dre_sys_load_snapshot.c

概要

システムコンテナをスナップショットの状態に復元するサンプル(“dre-sys load -n <name> -g <tag>”に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result:
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL
10          char *tag = NULL;
            ・・・(略)・・・
24          /* open handler */
25          h = dre_sys_open(name);
26          if(!h) {
27                  result = DRE_ERR_NULL_POINTER;
28                  goto error;
29          }
30
31          /* load snapshot */
32          result = dre_sys_load_snapshot(h, snapshot);
33          if(result) goto error;
34
35          /* close handler */
36          result = dre_sys_close(h);
37          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_load_snapshot.c を参照。)

25 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。32 行目の dre_sys_load_snapshot()で、<name>をスナップショット<tag>で復元します。36 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_load_snapshot　-n <name>　-t <tag>

引数

<name>	コンテナ名
<tag>	スナップショット名

1.2.2.16　dre_sys_migration_start.c

概要

システムコンテナのマイグレーションを開始するサンプル(“dre-sys migrate -n <name> -d <dest> -p <port>”に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          int port;
10          char *name = NULL;
11          char *dest = NULL;
            ・・・(略)・・・

27          /* open handler */
28          h = dre_sys_open(name);
29          if(!h) {
30                  result = DRE_ERR_NULL_POINTER;
31                  goto error;
32          }
33
34          /* start migration */
35          result = dre_sys_migration_start(h, dest, port);
36          if(result) goto error;
37
38          /* close handler */
39          result = dre_sys_close(h);
40          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_migration_start.c を参照。)

28 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。35 行目の dre_sys_migration_start()で、<name>のマイグレーションを開始します。39 行目の dre_sys_close()で、<name>のハンドルを解放します。マイグレーションを実行する際には、予めマイグレーション先で immigrate を実行しておく必要があります。

使用方法

dre_sys_migration_start　-n <name>　-d <dest>　--port <port>

引数

<name>	コンテナ名
<dest>	マイグレーション先のホスト名または IP アドレス
<port>	マイグレーションに使用するポート番号

1.2.2.17　dre_sys_remove_base.c

概要

ベースイメージを削除するサンプル(“dre-sys remove --base <base_image>”に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h;
 9          char *base = NULL;
            ・・・(略)・・・
21          /* remove base image */
22          result = dre_sys_remove_base(base);
23          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_remove_base.c を参照。)

22 行目の dre_sys_remove_base()で、ベースイメージ<base_image>を破棄します。

使用方法

dre_sys_remove_base　-b <base_image>

引数

<base_image_name>　　ベースイメージ名

1.2.2.18　dre_sys_remove_checkpoint.c

概要

システムコンテナのチェックポイントを削除するサンプル(“dre-sys remove --checkpoint -n <name> -t <tag>”に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
10          char *tag = NULL;
            ・・・(略)・・・
24          /* open handler */
25          h = dre_sys_open(name);
26          if(!h) {
27                  result = DRE_ERR_NULL_POINTER;
28                  goto error;
```

```
29        }
30
31        /* remove snapshot */
32        result = dre_sys_remove_checkpoint(h, snapshot);
33        if(result) goto error;
34
35        /* close handler */
36        result = dre_sys_close(h);
37        if(result) goto error;
          ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_remove_checkpoint.c を参照。)

25 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。32 行目の dre_sys_remove_checkpoint()で、<name>のチェックポイント<tag>を削除します。36 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_remove_checkpoint　-n <name>　-t <tag>

引数

<name>	コンテナ名
<tag>	スナップショット名

1.2.2.19　dre_sys_remove_snapshot.c

概要

システムコンテナのスナップショットを削除するサンプル(“dre-sys remove --snapshot -n <name> -t <tag>”に相当)

ソースコード解説

```
3   #include “dre_sys.h”
4
5   int main(int argc, char **argv)
6   {
7         dre_return_t result;
8         dre_sys_handle *h = NULL;
9         char *name = NULL;
10        char *tag = NULL;
          ・・・(略)・・・
24        /* open handler */
25        h = dre_sys_open(name);
26        if(!h) {
27              result = DRE_ERR_NULL_POINTER;
28              goto error;
29        }
30
31        /* remove snapshot */
```

```
32        result = dre_sys_remove_snapshot(h, snapshot);
33        if(result) goto error;
34
35        /* close handler */
36        result = dre_sys_close(h);
37        if(result) goto error;
          ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_remove_snapshot.c を参照。)

25 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。32 行目の dre_sys_remove_snapshot()で、<name>のスナップショット<tag>を削除します。36 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_remove_snapshot　-n <name> -t <tag>

引数

<name>	コンテナ名
<tag>	スナップショット名

1.2.2.20　dre_sys_restart.c

概要

システムコンテナの動作をチェックポイントから再開するサンプル(“dre-sys restart -n <name> -t <tag>”に相当)

ソースコード解説

```
3   #include “dre_sys.h”
4
5   int main(int argc, char **argv)
6   {
7         dre_return_t result:
8         dre_sys_handle *h = NULL;
9         char *name = NULL;
10        char *tag = NULL;
          ・・・(略)・・・
24        /* open handler */
25        h = dre_sys_open(name);
26        if(!h) {
27              result = DRE_ERR_NULL_POINTER;
28              goto error;
29        }
30
31        /* restart from checkpoint */
32        result = dre_sys_restart(h, tag);
33        if(result) goto error;
34
```

```
35        /* close handler */
36        result = dre_sys_close(h);
37        if(result) goto error;
          ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_restart.c を参照。)

25 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。32 行目の dre_sys_restart()で、<name>の動作をチェックポイント<tag>の状態から再開します。36 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_restart　-n <name>　-t <tag>

引数

<name>	コンテナ名
<tag>	チェックポイント名

1.2.2.21　dre_sys_resume.c

概要

一時停止中のシステムコンテナの動作を再開するサンプル(“dre-sys resume -n <name>”に相当)

ソースコード解説

```
3   #include “dre_sys.h”
4
5   int main(int argc, char **argv)
6   {
7         dre_return_t result;
8         dre_sys_handle *h = NULL;
9         char *name = NULL;
          ・・・(略)・・・
21        /* open handler */
22        h = dre_sys_open(name);
23        if(!h) {
24              result = DRE_ERR_NULL_POINTER;
25              goto error;
26        }
27
28        /* resume container */
29        result = dre_sys_resume(h);
30        if(result) goto error;
31
32        /* close handler */
33        result = dre_sys_close(h);
34        if(result) goto error;
          ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_resume.c を参照。)

22 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。29 行目の dre_sys_resume()で、<name>の動作を再開します。33 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_resume　-n <name>

引数

<name>	コンテナ名

1.2.2.22　dre_sys_save_snapshot.c

概要

システムコンテナのスナップショットを作成するサンプル(“dre-sys save -n <name> -t <tag>”に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
10          char *tag = NULL;
            ・・・(略)・・・
24          /* open handler */
25          h = dre_sys_open(name);
26          if(!h) {
27                  result = DRE_ERR_NULL_POINTER;
28                  goto error;
29          }
30
31          /* save snapshot */
32          result = dre_sys_save_snapshot(h, snapshot);
33          if(result) goto error;
34
35          /* close handler */
36          result = dre_sys_close(h);
37          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_save_snapshot.c を参照。)

25 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。32 行目の dre_sys_save_snapshot()で、<name>のスナップショット<tag>を作成します。36 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_save_snapshot　-n <name>　-t <tag>

引数

<name>	コンテナ名
<tag>	スナップショット名

1.2.2.23　dre_sys_set_resource.c

概要

システムコンテナが使用するリソースに制限をかけるサンプル(“dre-sys limit <resource_type> <limitation> -n <name> [-g <cgroup>] [-a <application_container>]“に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_sys_resource_value_t resource;
 8          dre_return_t result:
 9          dre_sys_handle *h = NULL;
10          char *name = NULL;
11          char *cgroup = NULL;
12          char *type = NULL;
13          char *app = NULL;
14          char *limitation = NULL;
            ・・・(略)・・・
42          /* set resource usage */
43          if(!strcmp(type, “cpu”)) {
44                  resource.type = DRE_SYS_CONTAINER_RESOURCE_CPU;
45                  resource.udata.cpu.percent = atoi(limitation);
46          } else if(!strcmp(type, “core”)) {
47                  resource.type = DRE_SYS_CONTAINER_RESOURCE_CPU_CORE;
48                  resource.udata.cpu.cores = limitation;
49          } else if(!strcmp(type, “mem”)) {
50                  resource.type = DRE_SYS_CONTAINER_RESOURCE_MEM;
51                  resource.udata.mem.usage_in_bytes = atoi(limitation);
52          } else if(!strcmp(type, “memsw”)) {
53                  resource.type = DRE_SYS_CONTAINER_RESOURCE_MEMSW;
54                  resource.udata.mem.memsw_usage_in_bytes = atoi(limitation);
55          } else {
56                  printf(“Invalid resource type: %s¥n”, type);
57                  result = DRE_ERR_INVALID_ARGUMENT;
58                  goto error;
59          }
60
61          /* open handler */
62          if(name) {
```

```
 63              h = dre_sys_open(name);
 64              if(!h) {
 65                      result = DRE_ERR_NULL_POINTER;
 66                      goto error;
 67              }
 68      }
 69
 70      /* set limitation */
 71      result = dre_sys_set_resource(h, cgroup, app, resource);
 72      if(result) goto error;
 73      printf("Set limitation of %s.¥n", type);
 74
 75      /* close handler */
 76      if(h) {
 77              result = dre_sys_close(h);
 78              if(result) goto error;
 79      }
         ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_set_resource.c を参照。)

43-59 行目で、制限を掛けるリソースと使用量を指定します。これは、実行時の引数によって以下の動作をします。

- <application_container>と<cgroup>を指定した場合は、システムコンテナ<name>上に存在するアプリケーションコンテナ<application_container>上のコントロールグループ<cgroup>が使用するリソースを指定します。
- <application_container>を省略し、<cgroup>を指定した場合は、<name>上のコントロールグループ<cgroup>が使用するリソースを指定します。
- <application_container>を指定し、<cgroup>を省略した場合は、<application_container>全体が使用するリソースを指定します。
- <application_container>と<cgroup>を省略した場合は、<name>全体のリソースを指定します。

63 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。71 行目の dre_sys_set_resource()で、43-59 行目で指定した内容でリソース使用量の制限を行います。
77 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

```
dre_sys_set_resource   -n <name>  <resource_type>  <limitation>  [-g <cgroup>]
                       [-a <application_container>]
```

引数

<name>	コンテナ名
<resource_type>	リソースの種類 “cpu”　:　CPU 使用率 “core”　:　CPU コア “mem”　:　メモリ使用量 “memsw”:　メモリとスワップ領域の使用量
<limitation>	リソース使用量の制限
<cgroup>	コントロールグループ(オプション)
<application_container>	アプリケーションコンテナ名(オプション)

1.2.2.24　dre_sys_share.c

概要

システムコンテナを NFS サーバと共有するサンプル(“dre-sys share -n <name>”に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
            ・・・(略)・・・
21          /* open handler */
22          h = dre_sys_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* share container */
29          result = dre_sys_share(h);
30          if(result) goto error;
31
32          /* close handler */
33          result = dre_sys_close(h);
34          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_share.c を参照。)

22 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。29 行目の dre_sys_share()で、<name>を NFS サーバと共有します。共有に使用する NFS サーバは、予め${DRE_NFS_PATH}に mount されている必要があります。(${DRE_NFS_PATH}については、/etc/default/dre を参照。) 33 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_share　-n <name>

引数

<name>	コンテナ名

1.2.2.25　dre_sys_start.c

概要

システムコンテナの動作を開始するサンプル(“dre-app start -n <name>”に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
            ・・・(略)・・・
21          /* open handler */
22          h = dre_sys_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* start container */
29          result = dre_sys_start(h);
30          if(result) goto error;
31
32          /* close handler */
33          result = dre_sys_close(h);
34          if(result) goto error;
            ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_start.c を参照。)

22 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。29 行目の dre_sys_start()で、<name>の動作を開始します。33 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_start　-n <name>

引数

<name>	コンテナ名

1.2.2.26　dre_sys_stop.c

概要

システムコンテナの動作を停止するサンプル(“dre-sys stop -n <name>”に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
```

```
 6 {
 7       dre_return_t result;
 8       dre_sys_handle *h = NULL;
 9       char *name = NULL;
         ・・・(略)・・・
21       /* open handler */
22       h = dre_sys_open(name);
23       if(!h) {
24             result = DRE_ERR_NULL_POINTER;
25             goto error;
26       }
27
28       /* stop container */
29       result = dre_sys_stop(h);
30       if(result) goto error;
31
32       /* close handler */
33       result = dre_sys_close(h);
34       if(result) goto error;
         ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_stop.c を参照。)

22 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。29 行目の dre_sys_stop()で、<name>の動作を停止します。33 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_stop　　-n <name>

引数

<name>	コンテナ名

1.2.2.27　dre_sys_suspend.c

概要

動作中のシステムコンテナを一時停止するサンプル(“dre-sys suspend -n <name>”に相当)

ソースコード解説

```
 3 #include “dre_sys.h”
 4
 5 int main(int argc, char **argv)
 6 {
 7       dre_return_t result;
 8       dre_sys_handle *h = NULL;
 9       char *name = NULL;
         ・・・(略)・・・
21       /* open handler */
22       h = dre_sys_open(name);
```

```
23      if(!h) {
24          result = DRE_ERR_NULL_POINTER;
25          goto error;
26      }
27
28      /* suspend container */
29      result = dre_sys_suspend(h);
30      if(result) goto error;
31
32      /* close handler */
33      result = dre_sys_close(h);
34      if(result) goto error;
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_suspend.c を参照。)

22 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。29 行目の dre_sys_suspend()で、<name>の動作を一時停止します。33 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_suspend　-n <name>

引数

<name>	コンテナ名

1.2.2.28　dre_sys_unbind_pid.c

概要

システムコンテナ上のコントロールグループからプロセスをアンバインドするサンプル(“dre-sys unbind -n <name> -g <cgroup> -pid <pid>”に相当)

ソースコード解説

```
3   #include “dre_sys.h”
4
5   int main(int argc, char **argv)
6   {
7       dre_return_t result;
8       dre_sys_handle *h = NULL;
9       char *name = NULL;
10      char *cgroup = NULL;
        ・・・(略)・・・
27      /* open handler */
28      if(name) {
29          h = dre_sys_open(name);
30          if(!h) {
31              result = DRE_ERR_NULL_POINTER;
32              goto error;
33          }
34      }
```

```
35
36      /* unbind from control group */
37      result = dre_sys_unbind_pid(h, cgroup, pid);
38      if(result) goto error;
39
40      /* close handler */
41      if(h) {
42          result = dre_sys_close(h);
43          if(result) goto error;
44      }
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_unbind_pid.c を参照。)

プログラムの引数でコンテナ名<name>を指定した場合、29 行目の dre_sys_open()で、<name>のハンドルを取得します。37 行目の dre_sys_unbind_pid()で、<pid>をコントロールグループ<cgroup>からアンバインドします。この時、<cgroup>は、<name>が指定された場合は<name>上のものが使用され、<name>を省略した場合はホスト OS 上のものが使用されます。<name>を指定した場合、42 行目の dre_sys_close()で<name>のハンドルを解放します。

使用方法

dre_sys_unbind_pid　　-g <cgroup>　--pid<pid>　[-n <name>]

引数

<cgroup>	コントロールグループ名
<pid>	プロセス ID
<name>	コンテナ名(オプション)

1.2.2.29　dre_sys_unset_resource.c

概要

システムコンテナのリソース制限を解除するサンプル("dre-sys unlimit <resource_type> -n <name> [-g <cgroup>] [-a <application_container>]"に相当)

ソースコード解説

```
3   #include "dre_sys.h"
4
5   int main(int argc, char **argv)
6   {
7       dre_sys_resource_value_t resource;
8       dre_return_t result;
9       dre_sys_handle *h = NULL;
10      char *name = NULL;
11      char *cgroup = NULL;
12      char *type = NULL;
13      char *app = NULL;
14      char *limitation = NULL;
        ・・・(略)・・・
```

```
37        /* unset resource usage */
38        if(!strcmp(type, "cpu")) {
39              resource.type = DRE_SYS_CONTAINER_RESOURCE_CPU;
40        } else if(!strcmp(type, "core")) {
41              resource.type = DRE_SYS_CONTAINER_RESOURCE_CPU_CORE;
42        } else if(!strcmp(type, "mem")) {
43              resource.type = DRE_SYS_CONTAINER_RESOURCE_MEM;
44        } else if(!strcmp(type, "memsw")) {
45              resource.type = DRE_SYS_CONTAINER_RESOURCE_MEMSW;
46        } else {
47              printf("Invalid resource type: %s¥n", type);
48              result = DRE_ERR_INVALID_ARGUMENT;
49              goto error;
50        }
51
52        /* open handler */
53        if(name) {
54              h = dre_sys_open(name);
55              if(!h) {
56                    result = DRE_ERR_NULL_POINTER;
57                    goto error;
58              }
59        }
60
61        /* unset limitation */
62        result = dre_sys_unset_resource(h, cgroup, app, resource);
63        if(result) goto error;
64        printf("Unset limitation of %s.¥n", type);
65
66        /* close handler */
67        if(h) {
68              result = dre_sys_close(h);
69              if(result) goto error;
70        }
          ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_unset_resource.c を参照。)

38-50 行目で、制限を解除するリソースを指定します。これは、実行時の引数によって以下の動作をします。

- <application_container>と<cgroup>を指定した場合は、システムコンテナ<name>上に存在するアプリケーションコンテナ<application_container>上のコントロールグループ<cgroup>が使用するリソースを指定します。
- <application_container>を省略し、<cgroup>を指定した場合は、<name>上のコントロールグループ<cgroup>が使用するリソースを指定します。
- <application_container>を指定し、<cgroup>を省略した場合は、<application_container>全体が使用するリソースを指定します。
- <application_container>と<cgroup>を省略した場合は、<name>全体のリソースを指定します。

54 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。62 行目の dre_sys_unset_resource()で、38-50 行目で指定した内容でリソース使用量の制限を解除します。
68 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_unset_resource　-n<name>　<resource_type>　[-g <cgroup>]　[-a <application_container>]

引数

<name>	コンテナ名
<resource_type>	リソースの種類
	“cpu”　:　CPU 使用率
	“core”　:　CPU コア
	“mem”　:　メモリ使用量
	“memsw”:　メモリとスワップ領域の使用量
<cgroup>	コントロールグループ(オプション)
<application_container>	アプリケーションコンテナ名(オプション)

1.2.2.30　dre_sys_unshare.c

概要

ローカルとNFSサーバにおけるシステムコンテナの共有を解除するサンプル(“dre-sys unshare -n <name>”に相当)

ソースコード解説

```
 3  #include “dre_sys.h”
 4
 5  int main(int argc, char **argv)
 6  {
 7          dre_return_t result;
 8          dre_sys_handle *h = NULL;
 9          char *name = NULL;
             ・・・(略)・・・
21          /* open handler */
22          h = dre_sys_open(name);
23          if(!h) {
24                  result = DRE_ERR_NULL_POINTER;
25                  goto error;
26          }
27
28          /* share container */
29          result = dre_sys_unshare(h);
30          if(result) goto error;
31
32          /* close handler */
33          result = dre_sys_close(h);
34          if(result) goto error;
             ・・・(略)・・・
```

（※ ソースコード全文は、/usr/share/doc/libdre0-dev/sample/sys/dre_sys_unshare.c を参照。）

22 行目の dre_sys_open()で、コンテナ<name>のハンドルを取得します。29 行目の dre_sys_unshare()で、ローカルとNFS サーバにおける<name>の共有を解除します。33 行目の dre_sys_close()で、<name>のハンドルを解放します。

使用方法

dre_sys_unshare　-n <name>

引数

<name>	コンテナ名

2 D-Aware Application API をコールするサンプルプログラム

2.1 サンプルプログラムの使用方法

libdaware0-dev をインストールすると、以下のパスに本 API を使用したサンプルプログラムのソースコードがインストールされます。

/usr/share/doc/libdaware0-dev/sample/

上記のパスで make コマンドを実行すると、サンプルプログラムがコンパイルできます。

2.2 サンプルプログラムの解説

以下のサンプルプログラムが付属します。

daware-termination.c

終了コールバック関数を登録・登録解除を行うサンプル

daware-log.c

フォーマットしたログを syslog へ出力するサンプル

2.2.1 daware-termination.c

概要

終了コールバック関数を登録・登録解除を行うサンプル

終了コールバックはシグナルを受信すると呼び出されます。
終了コールバック呼出し後に終了処理を行いプログラムが終了されることを想定しています。

ソースコード解説

```
 3  #include "daware.h"
    ・・・(略)・・・
 7  int termination_flag = 0;
 8
 9  int
10  termination_callback (int signo, void *arg)
11  {
12          termination_flag = 1;
13          return 0;
14  }
15
16  int
17  main (int argc, char *argv[])
18  {
            ・・・(略)・・・
21          DTermination *dterm = daware_termination_register_callback
22          (SIGTERM, termination_callback, NULL);
            ・・・(略)・・・
```

```
33      while (1) {
34              printf("%d¥n", val);
35              if (termination_flag) break;
36                      val++;
37              sleep(1);
38      }
39
40      fp = fopen(DATA_FILE, "wb");
41      if (fp) {
42              printf("save: %d > %s¥n", val, DATA_FILE);
43              fwrite(&val, sizeof(val), 1, fp);
44              fclose(fp);
45      }
46
47      daware_termination_unregister_callback(dterm);
        ・・・(略)・・・
```

(※ ソースコード全文は、/usr/share/doc/libdaware0-dev/sample/daware-termination.c を参照。)

21 行目では daware_termination_register_callback()関数にて終了コールバック関数の登録を行っています。ここでは SIGTERM シグナルを受信すると termination_callback()関数が呼び出されるように設定しています。

SIGTERM シグナルを受信すると 10 行目の termination_callback()関数が呼び出され、termination_flag に 1 が設定されます。

33-38 行目のループ内の 35 行目では termination_flag を監視しており、termination_flag が 1 になるとループ処理を抜け、40-45 行目の終了処理を実行します。

47 行目では daware_termination_unregister_callback()関数にて終了コールバック関数の登録解除を行っています。

使用方法

daware-termination を実行し、別ターミナルから kill コマンドでプロセスを終了させてください。

```
$ ./daware-termination
PID: 7217
0
1
2
3
save: 4 > /tmp/daware-termination.dat
```

```
$ kill 7217
```

上記の例では「3」を出力後に kill が実行され、終了処理（データを/tmp/daware-termination.dat に保存）が行われています。

2.2.2daware-log.c

概要

フォーマットしたログを syslog へ出力するサンプル

ソースコード解説

```
 1  #include <stdio.h>
 2  #include "daware.h"
 3
 4  int main (int argc, char *argv[])
 5  {
 6          const char format[] = "%faci%.%lvl% %file% line %line%: %msg%";
 7
 8          daware_log_open(argv[0], LOG_CONS|LOG_PID, LOG_LOCAL7);
 9
10          daware_log_set_format(format);
11
12          DLOG_NOTICE("message 1");
13          DLOG_INFO("message 2");
14          DLOG_DEBUG("message 3");
15
16          DLOG(LOG_USER|LOG_DEBUG, "message 4");
17
18          daware_log_close();
19
20          return 0;
21  }
```

(※ ソースコード全文は、/usr/share/doc/libdaware0-dev/sample/daware-log.c を参照。)

8 行目では syslog のオプション及び facility を設定し syslog との接続を開始しています。
6 行目と 10 行目ではログのフォーマットを設定しています。%faci%と%lvl%の箇所には syslog の facility と level、%file%にはファイル名、%line%には行番号、%msg%にはログメッセージが syslog に出力されます。
12-14 行目ではそれぞれ、syslog の level に応じたマクロを呼び出しています。
16 行目の DLOG マクロは facility と level が設定可能なマクロです。ここでは facility を LOG_USER、level を LOG_DEBUG に設定しています。
18 行目では syslog との接続を切断しています。

使用方法

daware-log を実行し、syslog への出力を確認してください。

```
$ ./daware-log
$ tail /var/log/syslog
Apr 17 16:36:55 ubuntu-12 ./daware-log[6977]: local5.notice daware-log.c line 12: message 1
Apr 17 16:36:55 ubuntu-12 ./daware-log[6977]: local5.info daware-log.c line 13: message 2
Apr 17 16:36:55 ubuntu-12 ./daware-log[6977]: local5.debug daware-log.c line 14: message 3
Apr 17 16:36:55 ubuntu-12 ./daware-log[6977]: user.debug daware-log.c line 16: message 4
```