Subject: [TOOL] BC: bcctl.c
Posted by Kirill Korotaev on Thu, 05 Oct 2006 15:59:20 GMT
View Forum Message <> Reply to Message

Utility (bcctl) usage and examples:
Usage: bcctl <cmd> [args...]
Commands:
    get
        gets current bc id
    enter <id> [<pid>]
        moves task into bc.
        if no pid - moves parent task
    stat <id> [<resource>]
        prints stats for bc's resource.
        if no resource - prints all resources
    set <id> <resource> <barrier> [<limit>]
        sets barrier/limit for bc's resource
        if no limit - it is set equal to barrier
Resources:
    kmemsize
    privvmpages
    physpages

Examples:
Move shell into beancounter
  $ bcctl enter 1
Look statistics
  $ bcctl stat 1
  resource          held    minheld    maxheld    barrier      limit    failcnt
  kmemsize           824          0       1056      65536      65536          0
  privvmpages        100          0        304 2147483647 2147483647          0
  physpages           40          0        150 2147483647 2147483647          0
See how reclamation work
  $ bcctl set 1 physpages 200
  $ memeat
  $ bcctl stat 1 physpages
  resource          held    minheld    maxheld    barrier      limit   failcnt
  physpages           40          0        200        200        200         0
Now turn swap off and run memeater again
  $ swapoff -a
  $ memeat
  Killed
  $ bcctl stat 1 physpages
  resource          held    minheld    maxheld    barrier      limit   failcnt
  physpages           41          0        200        200        200         1

---

```
#include <sys/syscall.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

typedef unsigned int bcid_t;

#include "bc/beancounter.h"

#define get_bcid()  (syscall(__NR_get_bcid))
#define set_bcid(id, pid) (syscall(__NR_set_bcid, id, pid))
#define set_bclimit(id, res, l) (syscall(__NR_set_bclimit, id, res, l))
#define get_bcstat(id, res, s) (syscall(__NR_get_bcstat, id, res, s))

static char *res_names[] = {
 "kmemsize",
 "privvmpages",
 "physpages",
 NULL,
};

static int bc_get_id(int argc, char **argv)
{
 bcid_t id;

 id = get_bcid();
 printf("%u\n", id);
 return 0;
}

static int bc_enter_id(int argc, char **argv)
{
 bcid_t id;
 pid_t pid;
 char *end;
 int ret;

 if (argc < 1)
  return -2;

 id = strtol(argv[0], &end, 10);
 if (*end != '\0')
  return -2;

 if (argc > 1) {
  pid = strtol(argv[1], &end, 10);
  if (*end != '\0')
```

```
   return -2;
 } else
  pid = getppid();

 ret = set_bcid(id, pid);
 if (ret < 0)
  perror("Can't set bcid");
 return ret;
}

static int bc_set_parms(int argc, char **argv)
{
 bcid_t id;
 int res;
 unsigned long limits[2];
 char *end;
 int ret;

 if (argc < 3)
  return -2;

 id = strtol(argv[0], &end, 10);
 if (*end != '\0')
  return -2;

 for (res = 0; res_names[res] != NULL; res++)
  if (strcmp(res_names[res], argv[1]) == 0)
   break;
 if (res_names[res] == NULL)
  return -2;

 limits[0] = strtoul(argv[2], &end, 10);
 if (*end != '\0')
  return -2;

 if (argc > 3) {
  limits[1] = strtoul(argv[3], &end, 10);
  if (*end != '\0')
   return -2;
 } else
  limits[1] = limits[0];

 ret = set_bclimit(id, res, limits);
 if (ret < 0)
  perror("Can't set limits");
 return ret;
}
```

```
static void bc_show_stat_hdr(void)
{
 printf("%-12s %10s %10s %10s %10s %10s %10s\n",
   "resource",
   "held",
   "minheld",
   "maxheld",
   "barrier",
   "limit",
   "failcnt");
}

static int bc_show_stat(bcid_t id, int res)
{
 struct bc_resource_parm parm;
 int ret;

 ret = get_bcstat(id, res, &parm);
 if (ret < 0) {
  perror("Can't get stat");
  return ret;
 }

 printf("%-12s %10lu %10lu %10lu %10lu %10lu %10lu\n",
   res_names[res],
   parm.held,
   parm.minheld,
   parm.maxheld,
   parm.barrier,
   parm.limit,
   parm.failcnt);
 return 0;
}

static int bc_get_stat(int argc, char **argv)
{
 bcid_t id;
 int res, ret;
 char *end;

 if (argc < 1)
  return -2;

 id = strtol(argv[0], &end, 10);
 if (*end != '\0')
  return -2;

 if (argc > 1) {
```

```
  for (res = 0; res_names[res] != NULL; res++)
   if (strcmp(res_names[res], argv[1]) == 0)
    break;
  if (res_names[res] == NULL)
   return -2;

  bc_show_stat_hdr();
  return bc_show_stat(id, res);
 }

 bc_show_stat_hdr();
 for (res = 0; res_names[res] != NULL; res++) {
  ret = bc_show_stat(id, res);
  if (ret < 0)
   return ret;
 }
 return 0;
}

static struct bc_command {
 char *name;
 int (*cmd)(int argc, char **argv);
 char *args;
 char *help;
} bc_commands[] = {
 {
  .name = "get",
  .cmd = bc_get_id,
  .args = "",
  .help = "gets current bc id",
 },
 {
  .name = "enter",
  .cmd = bc_enter_id,
  .args = "<id> [<pid>]",
  .help = "moves task into bc.\n"
    "\t\tif no pid - moves parent task",
 },
 {
  .name = "stat",
  .cmd = bc_get_stat,
  .args = "<id> [<resource>]",
  .help = "prints stats for bc's resource.\n\t\t"
    "if no resource - prints all resources",
 },
 {
  .name = "set",
  .cmd = bc_set_parms,
```

```
  .args = "<id> <resource> <barrier> [<limit>]",
  .help = "sets barrier/limit for bc's resource\n"
    "\t\tif no limit - it is set equal to barrier",
 },
 {
  .name = NULL,
 },
};

static void usage(void)
{
 int i;

 printf("Usage: bcctl <cmd> [args...]\n");
 printf("Commands:\n");
 for (i = 0; bc_commands[i].name != NULL; i++)
  printf("\t%s %s\n\t\t%s\n",
    bc_commands[i].name,
    bc_commands[i].args,
    bc_commands[i].help);
 printf("Resources:\n");
 for (i = 0; res_names[i] != NULL; i++)
  printf("\t%s\n", res_names[i]);
}

int main(int argc, char **argv)
{
 int i, ret;

 if (argc < 2) {
  usage();
  return 0;
 }

 ret = -1;
 for (i = 0; bc_commands[i].name != NULL; i++) {
  if (strcmp(argv[1], bc_commands[i].name) != 0)
   continue;

  ret = bc_commands[i].cmd(argc - 2, argv + 2);
 }

 if (ret < 0)
  usage();
 return ret;
}
```