

```

*****
21040 Wed Apr 16 10:22:21 2014
new/usr/src/cmd/cpc/common/cpustat.c
1100 cpustat usage message is incorrect
*****
_____unchanged_portion_omitted_____

107 static int cpustat(void);
108 static int get_chipid(kstat_ctl_t *kc, processorid_t cpuid);
109 static void *soaker(void *arg);

112 #if !defined(TEXT_DOMAIN)
113 #define TEXT_DOMAIN "SYS_TEST"
114 #endif

116 int
117 main(int argc, char *argv[])
118 {
119     struct options *opts = &__options;
120     int c, errcnt = 0, ret;
121     cpc_setgrp_t *sgrp;
122     char *errstr;
123     double period;
124     char *endp;
125     struct rlimit rl;

127     (void) setlocale(LC_ALL, "");
128     (void) textdomain(TEXT_DOMAIN);

130     if ((opts->pgmname = strrchr(argv[0], '/')) == NULL)
131         opts->pgmname = argv[0];
132     else
133         opts->pgmname++;

135     /* Make sure we can open enough files */
136     rl.rlim_max = rl.rlim_cur = RLIM_INFINITY;
137     if (setrlimit(RLIMIT_NOFILE, &rl) != 0) {
138         errstr = strerror(errno);
139         (void) fprintf(stderr,
140             gettext("%s: setrlimit failed - %s\n"),
141             opts->pgmname, errstr);
142     }

144     if ((cpc = cpc_open(CPC_VER_CURRENT)) == NULL) {
145         errstr = strerror(errno);
146         (void) fprintf(stderr, gettext("%s: cannot access performance "
147             "counters - %s\n"), opts->pgmname, errstr);
148         return (1);
149     }

151     (void) cpc_seterrhndlr(cpc, cpustat_errfn);
152     strtoset_errfn = cpustat_errfn;

154     /*
155      * Check to see if cpustat needs to be SMT-aware.
156      */
157     smt = smt_limited_cpc_hw(cpc);

159     /*
160      * Establish some defaults
161      */
162     opts->mseconds = 5000;
163     opts->nsamples = UINT_MAX;
164     opts->dotitle = 1;
165     if ((opts->master = cpc_setgrp_new(cpc, smt)) == NULL) {

```

```

166         (void) fprintf(stderr, gettext("%s: out of heap\n"),
167             opts->pgmname);
168         return (1);
169     }

171     while ((c = getopt(argc, argv, "Dc:hntT:sp:")) != EOF && errcnt == 0)
172         switch (c) {
173             case 'D': /* enable debugging */
174                 opts->debug++;
175                 break;
176             case 'c': /* specify statistics */
177                 if ((sgrp = cpc_setgrp_newset(opts->master,
178                     optarg, &errcnt)) != NULL)
179                     opts->master = sgrp;
180                 break;
181             case 'n': /* no titles */
182                 opts->dotitle = 0;
183                 break;
184             case 'p': /* periodic behavior */
185                 opts->doperiod = 1;
186                 period = strtod(optarg, &endp);
187                 if (*endp != '\0') {
188                     (void) fprintf(stderr, gettext("%s: invalid "
189                         "parameter \"%s\"\n"), opts->pgmname,
190                         optarg);
191                     errcnt++;
192                 }
193                 break;
194             case 's': /* run soaker thread */
195                 opts->dosoaker = 1;
196                 break;
197             case 't': /* print %tick */
198                 opts->dotick = 1;
199                 break;
200             case 'T':
201                 if (optarg) {
202                     if (*optarg == 'u')
203                         timestamp_fmt = UDATE;
204                     else if (*optarg == 'd')
205                         timestamp_fmt = DDATE;
206                     else
207                         errcnt++;
208                 } else {
209                     errcnt++;
210                 }
211                 break;
212             case 'h': /* help */
213                 opts->dohelp = 1;
214                 break;
215             case '?':
216             default:
217                 errcnt++;
218                 break;
219         }

221     switch (argc - optind) {
222     case 0:
223         break;
224     case 2:
225         opts->nsamples = strtol(argv[optind + 1], &endp, 10);
226         if (*endp != '\0') {
227             (void) fprintf(stderr,
228                 gettext("%s: invalid argument \"%s\"\n"),
229                 opts->pgmname, argv[optind + 1]);
230             errcnt++;
231             break;

```

```

232     }
233     /*FALLTHROUGH*/
234     case 1:
235         opts->mseconds = (uint_t)(strtod(argv[optind], &endp) * 1000.0);
236         if (*endp != '\0') {
237             (void) fprintf(stderr,
238                 gettext("%s: invalid argument \"%s\"\n"),
239                 opts->pgmname, argv[optind]);
240             errcnt++;
241         }
242         break;
243     default:
244         errcnt++;
245         break;
246     }

248     if (opts->nsamples == 0 || opts->mseconds == 0)
249         errcnt++;

251     if (errcnt != 0 || opts->dohelp ||
252         (opts->nsets = cpc_setgrp_numsets(opts->master)) == 0) {
253         (void) fprintf(opts->dohelp ? stdout : stderr, gettext(
254             "Usage:\n\t%s -c events [-c events]... [-p period] [-nstD] "
255             "[-T d|u] [interval [count]]\n\t%s -h\n\n"
256             "Usage:\n\t%s [-c events] [-p period] [-nstD] "
257             "[-T d|u] [interval [count]]\n\n"
258             "\t-c events specify processor events to be monitored\n"
259             "\t-n\t suppress titles\n"
260             "\t-p period cycle through event list periodically\n"
261             "\t-s\t run user soaker thread for system-only events\n"
262             "\t-t\t include %s register\n"
263             "\t-T d|u\t Display a timestamp in date (d) or unix "
264             "time_t (u)\n"
265             "\t-D\t enable debug mode\n"
266             "\t-h\t print extended usage information\n\n"
267             "\tUse cputrack(1) to monitor per-process statistics.\n"),
268             opts->pgmname, opts->pgmname, CPC_TICKREG_NAME);
269         (void) putchar('\n');
270         (void) capabilities(cpc, stdout);
271         exit(0);
272     }
273     exit(2);

275     /*
276     * If the user requested periodic behavior, calculate the rest time
277     * between cycles.
278     */
279     if (opts->doperiod) {
280         opts->mseconds_rest = (uint_t)((period * 1000.0) -
281             (opts->mseconds * opts->nsets));
282         if ((int)opts->mseconds_rest < 0)
283             opts->mseconds_rest = 0;
284         if (opts->nsamples != UINT_MAX)
285             opts->nsamples *= opts->nsets;
286     }

288     cpc_setgrp_reset(opts->master);
289     (void) setvbuf(stdout, NULL, _IOLBF, 0);

291     /*
292     * If no system-mode only sets were created, no soaker threads will be
293     * needed.
294     */

```

```

295         if (opts->dosoaker == 1 && cpc_setgrp_has_sysonly(opts->master) == 0)
296             opts->dosoaker = 0;

298         ret = cpustat();

300         (void) cpc_close(cpc);

302         return (ret);
303     }

```

---

unchanged portion omitted