```
*****************************************************************
   21018 Tue Feb  3 05:15:47 2015
new/usr/src/cmd/cpc/common/cpustat.c
1100 cpustat usage message is incorrect
*****************************************************************
______unchanged_portion_omitted_

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


#if !defined(TEXT_DOMAIN)
#define TEXT_DOMAIN     "SYS_TEST"
#endif

int
main(int argc, char *argv[])
{
        struct options  *opts = &__options;
        int             c, errcnt = 0, ret;
        cpc_setgrp_t    *sgrp;
        char            *errstr;
        double          period;
        char            *endp;
        struct rlimit   rl;

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

        if ((opts->pgmname = strrchr(argv[0], ’/’)) == NULL)
                opts->pgmname = argv[0];
        else
                opts->pgmname++;

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

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

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

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

        /*
         * Establish some defaults
         */
        opts->mseconds = 5000;
        opts->nsamples = UINT_MAX;
        opts->dotitle = 1;
        if ((opts->master = cpc_setgrp_new(cpc, smt)) == NULL) {
```

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

        while ((c = getopt(argc, argv, "Dc:hntT:sp:")) != EOF && errcnt == 0)
                switch (c) {
                case ’D’:                       /* enable debugging */
                        opts->debug++;
                        break;
                case ’c’:                       /* specify statistics */
                        if ((sgrp = cpc_setgrp_newset(opts->master,
                            optarg, &errcnt)) != NULL)
                                opts->master = sgrp;
                        break;
                case ’n’:                       /* no titles */
                        opts->dotitle = 0;
                        break;
                case ’p’:                       /* periodic behavior */
                        opts->doperiod = 1;
                        period = strtod(optarg, &endp);
                        if (*endp != ’\0’) {
                                (void) fprintf(stderr, gettext("%s: invalid "
                                    "parameter \"%s\"\n"), opts->pgmname,
                                    optarg);
                                errcnt++;
                        }
                        break;
                case ’s’:                       /* run soaker thread */
                        opts->dosoaker = 1;
                        break;
                case ’t’:                       /* print %tick */
                        opts->dotick = 1;
                        break;
                case ’T’:
                        if (optarg) {
                                if (*optarg == ’u’)
                                        timestamp_fmt = UDATE;
                                else if (*optarg == ’d’)
                                        timestamp_fmt = DDATE;
                                else
                                        errcnt++;
                        } else {
                                errcnt++;
                        }
                        break;
                case ’h’:                       /* help */
                        opts->dohelp = 1;
                        break;
                case ’?’:
                default:
                        errcnt++;
                        break;
                }

        switch (argc - optind) {
        case 0:
                break;
        case 2:
                opts->nsamples = strtol(argv[optind + 1], &endp, 10);
                if (*endp != ’\0’) {
                        (void) fprintf(stderr,
                            gettext("%s: invalid argument \"%s\"\n"),
                            opts->pgmname, argv[optind + 1]);
                        errcnt++;
                        break;
                }
                /*FALLTHROUGH*/
        case 1:
                opts->mseconds = (uint_t)(strtod(argv[optind], &endp) * 1000.0);
                if (*endp != '\0') {
                        (void) fprintf(stderr,
                            gettext("%s: invalid argument \"%s\"\n"),
                            opts->pgmname, argv[optind]);
                        errcnt++;
                }
                break;
        default:
                errcnt++;
                break;
        }

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

        if (errcnt != 0 || opts->dohelp ||
            (opts->nsets = cpc_setgrp_numsets(opts->master)) == 0) {
                (void) fprintf(opts->dohelp ? stdout : stderr, gettext(
                    "Usage:\n\t%s -c spec [-c spec]... [-p period] [-T u|d]\n"
                    "\t\t[-sntD] [interval [count]]\n\n"
                    "\t-c spec\t  specify processor events to be monitored\n"
                    "Usage:\n\t%s [-c events] [-p period] [-nstD] "
                    "[-T d|u] [interval [count]]\n\n"
                    "\t-c events specify processor events to be monitored\n"
                    "\t-n\t  suppress titles\n"
                    "\t-p period cycle through event list periodically\n"
                    "\t-s\t  run user soaker thread for system-only events\n"
                    "\t-t\t  include %s register\n"
                    "\t-T d|u\t  Display a timestamp in date (d) or unix "
                    "time_t (u)\n"
                    "\t-D\t  enable debug mode\n"
                    "\t-h\t  print extended usage information\n\n"
                    "\tUse cputrack(1) to monitor per-process statistics.\n"),
                    opts->pgmname, CPC_TICKREG_NAME);
                if (opts->dohelp) {
                        (void) putchar('\n');
                        (void) capabilities(cpc, stdout);
                        exit(0);
                }
                exit(2);
        }

        /*
         * If the user requested periodic behavior, calculate the rest time
         * between cycles.
         */
        if (opts->doperiod) {
                opts->mseconds_rest = (uint_t)((period * 1000.0) -
                    (opts->mseconds * opts->nsets));
                if ((int)opts->mseconds_rest < 0)
                        opts->mseconds_rest = 0;
                if (opts->nsamples != UINT_MAX)
                        opts->nsamples *= opts->nsets;
        }

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

        /*
         * If no system-mode only sets were created, no soaker threads will be
         * needed.
         */
```

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

        ret = cpustat();

        (void) cpc_close(cpc);

        return (ret);
}
```

______***unchanged_portion_omitted_***