```
**********************************************************
   26807 Thu Jan 24 10:07:37 2019
new/usr/src/cmd/stat/common/acquire_iodevs.c
10139 smatch fixes for usr/src/cmd/stat
**********************************************************
______unchanged_portion_omitted_

 399 /* select which I/O devices to collect stats for */
 400 static void
 401 choose_iodevs(struct snapshot *ss, struct iodev_snapshot *iodevs,
 402     struct iodev_filter *df)
 403 {
 404         struct iodev_snapshot   *pos, *ppos, *tmp, *ptmp;
 405         int                     nr_iodevs;
 406         int                     nr_iodevs_orig;

 408         nr_iodevs = df ? df->if_max_iodevs : UNLIMITED_IODEVS;
 409         nr_iodevs_orig = nr_iodevs;

 411         if (nr_iodevs == UNLIMITED_IODEVS)
 412                 nr_iodevs = INT_MAX;

 414         /* add the full matches */
 415         pos = iodevs;
 416         while (pos && nr_iodevs) {
 417                 tmp = pos;
 418                 pos = pos->is_next;

 420                 if (!iodev_match(tmp, df))
 421                         continue;       /* failed full match */

 423                 list_del(&iodevs, tmp);
 424                 insert_iodev(ss, tmp);

 426                 /*
 427                  * Add all mpxio paths associated with match above. Added
 428                  * paths don't count against nr_iodevs.
 429                  */
 430                 if (strchr(tmp->is_name, '.') == NULL) {
 431                 ppos = iodevs;
 432                 while (ppos) {
 433                         ptmp = ppos;
 434                         ppos = ppos->is_next;

 436                         if (!iodev_path_match(tmp, ptmp))
 437                                 continue;       /* not an mpxio path */

 439                         list_del(&iodevs, ptmp);
 440                         insert_iodev(ss, ptmp);
 441                         if (pos == ptmp)
 442                                 pos = ppos;
 443                 }
 444                 }

 446                 nr_iodevs--;
 447         }

 449         /*
 450          * If we had a filter, and *nothing* passed the filter then we
 451          * don't want to fill the  remaining slots - it is just confusing
 452          * if we don that, it makes it look like the filter code is broken.
 453          */
 454         if ((df->if_nr_names == NULL) || (nr_iodevs != nr_iodevs_orig)) {
 455                 /* now insert any iodevs into the remaining slots */
 456                 pos = iodevs;
 457                 while (pos && nr_iodevs) {
```



```
 458                         tmp = pos;
 459                         pos = pos->is_next;

 461                         if (df->if_skip_floppy &&
 461                         if (df && df->if_skip_floppy &&
 462                             strncmp(tmp->is_name, "fd", 2) == 0)
 463                                 continue;

 465                         list_del(&iodevs, tmp);
 466                         insert_iodev(ss, tmp);

 468                         --nr_iodevs;
 469                 }
 470         }

 472         /* clear the unwanted ones */
 473         pos = iodevs;
 474         while (pos) {
 475                 struct iodev_snapshot *tmp = pos;
 476                 pos = pos->is_next;
 477                 free_iodev(tmp);
 478         }
 479 }
______unchanged_portion_omitted_
```

```
**********************************************************
   13692 Thu Jan 24 10:07:37 2019
new/usr/src/cmd/stat/vmstat/vmstat.c
10139 smatch fixes for usr/src/cmd/stat
**********************************************************
```

```
/*
 * Copyright (c) 1991, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/*
 * Copyright (c) 1980 Regents of the University of California.
 * All rights reserved.  The Berkeley software License Agreement
 * specifies the terms and conditions for redistribution.
 */

/* from UCB 5.4 5/17/86 */
/* from SunOS 4.1, SID 1.31 */

/*
 * Copyright (c) 2018, Joyent, Inc.
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <ctype.h>
#include <unistd.h>
#include <memory.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>
#include <signal.h>
#include <values.h>
#include <poll.h>
#include <locale.h>

#include "statcommon.h"

char *cmdname = "vmstat";
int caught_cont = 0;

static uint_t timestamp_fmt = NODATE;

static  int     hz;
static  int     pagesize;
static  double  etime;
static  int     lines = 1;
static  int     swflag = 0, pflag = 0;
static  int     suppress_state;
static  long    iter = 0;
static  hrtime_t period_n = 0;
static  struct  snapshot *ss;

struct iodev_filter df;

#define pgtok(a) ((a) * (pagesize >> 10))
#define denom(x) ((x) ? (x) : 1)
#define REPRINT 19

static  void    dovmstats(struct snapshot *old, struct snapshot *new);
static  void    printhdr(int);
static  void    dosum(struct sys_snapshot *ss);
static  void    dointr(struct snapshot *ss);
static  void    usage(void);

int
```



```
main(int argc, char **argv)
{
        struct snapshot *old = NULL;
        enum snapshot_types types = SNAP_SYSTEM;
        int summary = 0;
        int intr = 0;
        kstat_ctl_t *kc;
        int forever = 0;
        hrtime_t start_n;
        int c;

        (void) setlocale(LC_ALL, "");
#if !defined(TEXT_DOMAIN)               /* Should be defined by cc -D */
#define TEXT_DOMAIN "SYS_TEST"          /* Use this only if it weren't */
#endif
        (void) textdomain(TEXT_DOMAIN);

        pagesize = sysconf(_SC_PAGESIZE);
        hz = sysconf(_SC_CLK_TCK);

        while ((c = getopt(argc, argv, "ipqsST:")) != EOF)
                switch (c) {
                case 'S':
                        swflag = !swflag;
                        break;
                case 's':
                        summary = 1;
                        break;
                case 'i':
                        intr = 1;
                        break;
                case 'q':
                        suppress_state = 1;
                        break;
                case 'p':
                        pflag++;        /* detailed paging info */
                        break;
                case 'T':
                        if (optarg) {
                                if (*optarg == 'u')
                                        timestamp_fmt = UDATE;
                                else if (*optarg == 'd')
                                        timestamp_fmt = DDATE;
                                else
                                        usage();
                        } else {
                                usage();
                        }
                        break;
                default:
                        usage();
                }

        argc -= optind;
        argv += optind;

        /* consistency with iostat */
        types |= SNAP_CPUS;

        if (intr)
                types |= SNAP_INTERRUPTS;
        if (!intr)
                types |= SNAP_IODEVS;

        /* max to fit in less than 80 characters */
        df.if_max_iodevs = 4;
```

```
128         df.if_allowed_types = IODEV_DISK;
129         df.if_nr_names = 0;
130         df.if_names = safe_alloc(df.if_max_iodevs * sizeof (char *));
131         (void) memset(df.if_names, 0, df.if_max_iodevs * sizeof (char *));

133         while (argc > 0 && !isdigit(argv[0][0]) &&
134             df.if_nr_names < df.if_max_iodevs) {
135                 df.if_names[df.if_nr_names] = *argv;
136                 df.if_nr_names++;
137                 argc--, argv++;
138         }

140         kc = open_kstat();

142         start_n = gethrtime();

144         ss = acquire_snapshot(kc, types, &df);

146         /* time, in seconds, since boot */
147         etime = ss->s_sys.ss_ticks / hz;

149         if (intr) {
150                 dointr(ss);
151                 free_snapshot(ss);
152                 exit(0);
153         }
154         if (summary) {
155                 dosum(&ss->s_sys);
156                 free_snapshot(ss);
157                 exit(0);
158         }

160         if (argc > 0) {
161                 long interval;
162                 char *endptr;

164                 errno = 0;
165                 interval = strtol(argv[0], &endptr, 10);

167                 if (errno > 0 || *endptr != '\0' || interval <= 0 ||
168                     interval > MAXINT)
169                         usage();
170                 period_n = (hrtime_t)interval * NANOSEC;
171                 if (period_n <= 0)
172                         usage();
173                 iter = MAXLONG;
174                 if (argc > 1) {
175                         iter = strtol(argv[1], NULL, 10);
176                         if (errno > 0 || *endptr != '\0' || iter <= 0)
177                                 usage();
178                 } else
179                         forever = 1;
180                 if (argc > 2)
181                         usage();
182         }

184         (void) sigset(SIGCONT, printhdr);

186         dovmstats(old, ss);
187         while (forever || --iter > 0) {
188                 /* (void) poll(NULL, 0, poll_interval); */

190                 /* Have a kip */
191                 sleep_until(&start_n, period_n, forever, &caught_cont);

193                 free_snapshot(old);
```



```
194                 old = ss;
195                 ss = acquire_snapshot(kc, types, &df);

197                 if (!suppress_state)
198                         snapshot_report_changes(old, ss);

200                 /* if config changed, show stats from boot */
201                 if (snapshot_has_changed(old, ss)) {
202                         free_snapshot(old);
203                         old = NULL;
204                 }

206                 dovmstats(old, ss);
207         }

209         free_snapshot(old);
210         free_snapshot(ss);
211         free(df.if_names);
212         (void) kstat_close(kc);
213         return (0);
214 }

216 #define DELTA(v) (new->v - (old ? old->v : 0))
217 #define ADJ(n)  ((adj <= 0) ? n : (adj >= n) ? 1 : n - adj)
218 #define adjprintf(fmt, n, val)  adj -= (n + 1) - printf(fmt, ADJ(n), val)

220 static int adj; /* number of excess columns */

222 /*ARGSUSED*/
223 static void
224 show_disk(void *v1, void *v2, void *d)
225 {
226         struct iodev_snapshot *old = (struct iodev_snapshot *)v1;
227         struct iodev_snapshot *new = (struct iodev_snapshot *)v2;
228         hrtime_t oldtime = new->is_crtime;
229         double hr_etime;
230         double reads, writes;

228         if (new == NULL)
229                 return;

232         if (old)
233                 oldtime = old->is_stats.wlastupdate;
234         hr_etime = new->is_stats.wlastupdate - oldtime;
235         if (hr_etime == 0.0)
236                 hr_etime = NANOSEC;
237         reads = new->is_stats.reads - (old ? old->is_stats.reads : 0);
238         writes = new->is_stats.writes - (old ? old->is_stats.writes : 0);
239         adjprintf(" %*.0f", 2, (reads + writes) / hr_etime * NANOSEC);
240 }
```

_____***unchanged_portion_omitted_***