```
**********************************************************
   18492 Thu Feb 12 20:33:34 2015
new/usr/src/uts/armv6/loader/fakeloader.c
loader: map as much as possible using 1MB pages
Chances are that we never actually executed this bit of code since all the
maps we ever deal with are either very short or much larger than 1MB.
**********************************************************
______unchanged_portion_omitted_

406 /*
407  * Finally, do all the dirty work. Let's create some page tables. The L1 page
408  * table is full of 1 MB mappings by default. The L2 Page table is 1k in size
409  * and covers that 1 MB. We're going to always create L2 page tables for now
410  * which will use 4k and 64k pages.
411  */
412 static void
413 fakeload_map(armpte_t *pt, uintptr_t pstart, uintptr_t vstart, size_t len,
414     uint32_t prot)
415 {
416         int entry, chunksize;
417         armpte_t *pte, *l2pt;
418         arm_l1pt_t *l1pt;

420         /*
421          * Make sure both pstart + vstart are 4k aligned, along with len.
422          */
423         if (pstart & MMU_PAGEOFFSET)
424                 fakeload_panic("pstart is not 4k aligned");
425         if (vstart & MMU_PAGEOFFSET)
426                 fakeload_panic("vstart is not 4k aligned");
427         if (len & MMU_PAGEOFFSET)
428                 fakeload_panic("len is not 4k aligned");

430         /*
431          * We're going to logically deal with each 1 MB chunk at a time.
432          */
433         while (len > 0) {
434                 if (vstart & MMU_PAGEOFFSET1M) {
435                         chunksize = MIN(len, MMU_PAGESIZE1M -
436                             (vstart & MMU_PAGEOFFSET1M));
437                 } else {
438                         chunksize = MIN(len, MMU_PAGESIZE1M);
439                 }

441                 entry = ARMPT_VADDR_TO_L1E(vstart);
442                 pte = &pt[entry];

444                 if (!ARMPT_L1E_ISVALID(*pte)) {
445                         uintptr_t l2table;

447                         if (!(vstart & MMU_PAGEOFFSET1M) &&
448                             !(pstart & MMU_PAGEOFFSET1M) &&
449                             len >= MMU_PAGESIZE1M) {
449                             len == MMU_PAGESIZE1M) {
450                                 fakeload_map_1mb(pstart, vstart, prot);
451                                 vstart += MMU_PAGESIZE1M;
452                                 pstart += MMU_PAGESIZE1M;
453                                 len -= MMU_PAGESIZE1M;
454                                 continue;
455                         }

457                         l2table = fakeload_alloc_l2pt();
458                         *pte = 0;
459                         l1pt = (arm_l1pt_t *)pte;
460                         l1pt->al_type = ARMPT_L1_TYPE_L2PT;
461                         l1pt->al_ptaddr = ARMPT_ADDR_TO_L1PTADDR(l2table);
```



```
462                 } else if ((*pte & ARMPT_L1_TYPE_MASK) != ARMPT_L1_TYPE_L2PT) {
463                         fakeload_panic("encountered l1 entry that's not a "
464                             "pointer to a level 2 table\n");
465                 } else {
466                         l1pt = (arm_l1pt_t *)pte;
467                 }

469                 /* Now that we have the l1pt fill in l2 entries */
470                 l2pt = (void *)(l1pt->al_ptaddr << ARMPT_L1PT_TO_L2_SHIFT);
471                 len -= chunksize;
472                 while (chunksize > 0) {
473                         arm_l2e_t *l2pte;

475                         entry = ARMPT_VADDR_TO_L2E(vstart);
476                         pte = &l2pt[entry];

478 #ifdef  MAP_DEBUG
479                         fakeload_puts("4k page pa->va, l2root, entry\n");
480                         fakeload_ultostr(pstart);
481                         fakeload_puts("->");
482                         fakeload_ultostr(vstart);
483                         fakeload_puts(", ");
484                         fakeload_ultostr((uintptr_t)l2pt);
485                         fakeload_puts(", ");
486                         fakeload_ultostr(entry);
487                         fakeload_puts("\n");
488 #endif

490                         if ((*pte & ARMPT_L2_TYPE_MASK) !=
491                             ARMPT_L2_TYPE_INVALID)
492                                 fakeload_panic("found existing l2 page table, "
493                                     "overlap in requested mappings detected!");
494                         /* Map vaddr to our paddr! */
495                         l2pte = ((arm_l2e_t *)pte);
496                         *pte = 0;
497                         if (!(prot & PF_X))
498                                 l2pte->ale_xn = 1;
499                         l2pte->ale_ident = 1;
500                         if (prot & PF_DEVICE) {
501                                 l2pte->ale_bbit = 1;
502                                 l2pte->ale_cbit = 0;
503                                 l2pte->ale_tex = 0;
504                                 l2pte->ale_sbit = 1;
505                         } else {
506                                 l2pte->ale_bbit = 1;
507                                 l2pte->ale_cbit = 1;
508                                 l2pte->ale_tex = 1;
509                                 l2pte->ale_sbit = 1;
510                         }
511                         if (prot & PF_W) {
512                                 l2pte->ale_ap2 = 1;
513                                 l2pte->ale_ap = 1;
514                         } else {
515                                 l2pte->ale_ap2 = 0;
516                                 l2pte->ale_ap = 1;
517                         }
518                         l2pte->ale_ngbit = 0;
519                         l2pte->ale_addr = ARMPT_PADDR_TO_L2ADDR(pstart);

521                         chunksize -= MMU_PAGESIZE;
522                         vstart += MMU_PAGESIZE;
523                         pstart += MMU_PAGESIZE;
524                 }
525         }
526 }
______unchanged_portion_omitted_
```