```
pgd_idx = pgd_index(PAGE_OFFSET); /* 3 */

/* the first 3 entries are for user space, and are pointing to the same empty_zero_page.*/
for (i=0; i<pgd_idx; i++)
  set_pgd(swapper_pg_dir + i, __pgd(__pa(empty_zero_page) + 0x001)); /* 0x001 == Present */

/* the 4th entry is for kernel space*/
pgd = swapper_pg_dir + pgd_idx;
phys_addr = 0x00000000;

/* i=3 initially. PTRS_PER_PGD=4 */
for (; i<PTRS_PER_PGD; ++i, ++pgd) {
  /* get the address of a PMD.
     The PMD maps 1G allocated by alloc_bootmem_low_pages() */
  pmd = (pmd_t *) alloc_bootmem_low_pages(PAGE_SIZE);
  /* the 4th entry is initialized with the above PMD */
  set_pgd(pgd, __pgd(__pa(pmd) | 0x001)); /* 0x001 == Present */

  if (phys_addr < max_low_pfn * PAGE_SIZE) /* cover ZONE_NORMAL */
    for (j=0; j < PTRS_PER_PMD /* 512 */
      && phys_addr < max_low_pfn*PAGE_SIZE; ++j) {
      /* fill up each PMD entry */
      set_pmd(pmd, __pmd(phys_addr | pgprot_val(__pgprot(0x1e3))));
      /* 0x1e3 == Present, Accessed, Dirty, Read/Write,
         Page Size, Global */

      /* each PMD entry covers 2M */
      phys_addr += PTRS_PER_PTE * PAGE_SIZE; /* 0x200000 */
    }
 }

/* The fourth Page Global Directory entry is then copied into the first entry, so as to
   mirror the mapping of the low physical memory in the first 896 MB of the linear address
   space. This mapping is required in order to complete the initialization of SMP systems:
   when it is no longer necessary, the kernel clears the corresponding page table entries
   by invoking the zap_low_mappings() function, as in the previous cases. */
swapper_pg_dir[0] = swapper_pg_dir[pgd_idx];
```