兼分析printk() / early_print() / early_printk()区别

in Documentation/kernel-parameters.txt

---

earlyprintk= [X86,SH,BLACKFIN,ARM,M68k]

earlyprintk=vga

earlyprintk=efi

earlyprintk=xen

earlyprintk=serial[,ttySn[,baudrate]]

earlyprintk=serial[,0x...[,baudrate]]

earlyprintk=ttySn[,baudrate]

earlyprintk=dbgp[debugController#]

earlyprintk is useful when the kernel crashes before

the normal console is initialized. It is not enabled by

default because it has some cosmetic problems.

Append ",keep" to not disable it when the real console

takes over.

Only one of vga, efi, serial, or usb debug port can

be used at a time.

Currently only ttyS0 and ttyS1 may be specified by

name. Other I/O ports may be explicitly specified

on some architectures (x86 and arm at least) by

replacing ttySn with an I/O port address, like this:

earlyprintk=serial,0x1008,115200

You can find the port for a given device in

/proc/tty/driver/serial:

2: uart:ST16650V2 port:00001008 irq:18 ...

Interaction with the standard serial driver is not

very good.

The VGA and EFI output is eventually overwritten by

the real console.

The xen output can only be used by Xen PV guests.

对ARM arch而言,在CONFIG_EARLY_PRINTK enable的情况下有early_printk console.

Gr2 / Gs2 LSP enable CONFIG_EARLY_PRINTK。

in arch/arm/kernel/early_printk.c

```
1.  extern void printch(int);
2.
3.  static void early_write(const char *s, unsigned n)
4.  {
5.          while (n-- > 0) {
6.                  if (*s == '\n')
7.                          printch('\r');
8.                  printch(*s);
9.                  s++;
10.         }
11. }
12. static void early_console_write(struct console *con, const char *s, unsigned n)
13. {
14.         early_write(s, n);
15. }
16. static struct console early_console_dev = {
17.         .name =         "earlycon",
18.         .write =        early_console_write,
19.         .flags =CON_PRINTBUFFER| CON_BOOT,
20.         .index =        -1,
21. };
22. static int __init setup_early_printk(char *buf)
23. {
24.         early_console = &early_console_dev;
25.         register_console(&early_console_dev);
26.         return 0;
27. }
28. early_param("earlyprintk", setup_early_printk);
```

如果在kernel parameter "earlyprintk" enable，则printk()也会通过early_console_write()

output。

early_console_write()其实就等同与arch/arm/kernel/setup.c中的early_print()，

就是直接写初始化好的串口。early_console_write()能work的条件同early_print()一样！

in kernel/printk/printk.c

```
#ifdef CONFIG_EARLY_PRINTK
struct console *early_console;

void early_vprintk(const char *fmt, va_list ap)
{
        if (early_console) {
                char buf[512];
                int n = vscnprintf(buf, sizeof(buf), fmt, ap);

                early_console->write(early_console, buf, n);
        }
}

asmlinkage __visible void early_printk(const char *fmt, ...)
{
        va_list ap;

        va_start(ap, fmt);
        early_vprintk(fmt, ap);
        va_end(ap);
}
#endif
```

early_print() 与 earlyprintk()之间的关系?

early_print()依赖于汇编function printch()，而earlyprintk()通过early_console->write()

实现，而early_console->write()还是依赖于printch()。

另外enable CONFIG_EARLY_PRINTK的一个好处，可能是在console_init()以前，printk()

function将有机会输出。

因为在setup_early_printk()会注册early_console_dev console,即在console_init()被调用

以前，已经有一个可以输出log的console了。否则的话，printk()的输出只能积累在log buffer中。

early_print()与early_printk()几乎是完全相同的，都是直接对uart io操作来输出log，

没有任何buffer，最可靠，当然效率也最低。同时在interrupt handler最好不要使用，但如果

使用的话，除了有点奇怪，我也想不出为什么不可以(比如你在一个timer interrupt handler去操作uart的register,固然有点奇怪，但也未尝不可)。

在early_console_dev console enable和console_init()被调用以前，printk()缓存log

并且可以实质性输出log(通过early_console_dev console)。