CaOS

(Calcium Operating System)

기타 유틸리티 함수

2007.9.5 v0.05 문서 작성

김기오

www.asmlove.co.kr

기타 유틸리티 함수

1. kprintf.c

caos_printf 함수는 printf 함수와 유사한 기능을 가지고 있다.

- escape sequence
* ₩n: 커서를 다음 줄로 넘김
* ₩b: 한 글자를 지움
- conversion specifier
* ₩x: 정수를 16진수로 출력
* ₩d: 정수를 10진수로 출력
* ₩s: 문자열 출력
* ₩c: 한 문자 출력

가변 인자를 처리하기 위해 types.h에 다음과 같은 매크로를 만들어야 한다.

```
typedef char *va_list;

#define va_start(ap, last) (void)((ap)=(va_list)&(last)+sizeof(last))


#define   _AUPBND                    (sizeof (int) - 1)
#define   _ADNBND                    (sizeof (int) - 1)
#define _bnd(X, bnd)                 (((sizeof (X)) + (bnd)) & (~(bnd)))
#define va_arg(ap, T)                (*(T *)(((ap) += (_bnd (T, _AUPBND))) - (_bnd (T,_ADNBND))))

#define va_end(ap) (void)((ap)=NULL
```

caos_printf() 함수는 kprintf.c 파일에 구현되었다.

```
int caos_printf(const char *format, ...)
{
                va_list ap;
                int i = 0;
```

```
		va_start(ap, format);
```

vs_start() 매크로는 여러 개의 인자를 구분해주고 caos_vsprintf() 함수로 넘긴다. sz는 conversion specifier를 처리한 결과를 저장하는 버퍼이다.

```
		caos_vsprintf(sz, format, ap);
		va_end(ap);
```

caos_vsprintf() 함수는 escape sequence를 처리하지 않는다. 따라서 caos_printf() 함수에서 처리한다.

```
		while (sz[i] != '₩0') {
			if (sz[i] == '₩n') {
				// move to next line
				set_cursor( ((int)(cursor_offset/80) + 1)*80 );
				i++;
				continue;
			} else if (sz[i] == '₩b') {
				caos_delchar(1);
				i++;
				continue;
			}
			caos_putchar(sz[i++]);
		}

		return i;
}


void caos_vsprintf(char *A_szString, const char *A_szFormat, va_list A_pAp)
{
		int temp;
```

conversion specifier에 따라 인자를 변환한다. va_arg() 매크로는 va_list 인자 리스트와 인자의 자료형을 지정하면 자료형에 맞게 인자를 반환해준다.

```
		while (*A_szFormat != '₩0') {
			// print decimal number
```

```
                if (*A_szFormat == '%' && *(A_szFormat+1) == 'd') {
```

va_list에서 int형으로 인자를 꺼낸다.

```
                        temp = va_arg(A_pAp, int);
```

int 형이므로 정수를 10진수 문자열로 변환해서 버퍼에 복사한다. print_decimal 함수는 변환된 문자열의 길이를 반환한다.

```
                        temp = print_decimal(A_szString, temp, *(A_szFormat+1));
                        A_szString += temp;
                        A_szFormat += 2;
                // print hexa-decimal number
                } else if (*A_szFormat == '%' && *(A_szFormat+1) == 'x' ) {
                        temp = va_arg(A_pAp, int);
                        temp = print_hex(A_szString, temp, *(A_szFormat+1));
                        A_szString += temp;
                        A_szFormat += 2;
                // print scring
                } else if (*A_szFormat == '%' && *(A_szFormat+1) == 's') {
                        temp = caos_strcat(A_szString, va_arg(A_pAp, char *));
                        A_szString += temp;
                        A_szFormat += 2;
                // print one character
                } else if (*A_szFormat == '%' && *(A_szFormat+1) == 'c') {
                        *A_szString++ = va_arg(A_pAp, char);
                        A_szFormat += 2;
                // normal character is copied as itself
                } else {
                        *A_szString++ = *A_szFormat++;
                }
        }
        *A_szString = '₩0';

}
```

2. io.c

인텔 프로세서에는 하드웨어 포트에 값을 쓰거나 읽기 위한 어셈블리 명령어가 따로 있다. 이런

어셈블리 명령어를 C 언어에서 호출하기 위해 wrapping function을 만들었다.
포트 번호를 인자로 전달하면 해당 포트에서 바이트 값을 읽고 반환한다.

```
unsigned char inb(unsigned short port)
{
        unsigned char _v;
        __asm__ __volatile__ ("inb %w1, %0":"=a" (_v):"Nd" (port));
        return _v;
}
```

포트 번호를 인자로 전달하면 해당 포트에서 워드 값을 읽고 반환한다.

```
unsigned short inw(unsigned short port)
{
        unsigned short _v;
        __asm__ __volatile__ ("inw %w1,%0":"=a" (_v):"Nd"(port));
        return _v;
}
```

포트 번호와 바이트 값을 전달하면 해당 포트에 값을 써넣는다.

```
void outb(unsigned short port, unsigned char value)
{
        __asm__ __volatile__ ("outb %b0,%w1": :"a" (value), "Nd"(port));
}
```

포트 번호와 워드 값을 전달하면 해당 포트에 값을 써넣는다.

```
void outw(unsigned short port, unsigned short value)
{
        __asm__ __volatile__ ("outw %w0,%w1": :"a" (value), "Nd"(port));
}
```