UCSC Extension – Linux Systems Programming
Homework #1

Jae Yang Park (jaeyangp@gmail.com)

1.1 Write a simple program that will run 100% user CPU time.

```
#include <stdio.h>

int main()
{
    while (1) ;
}
```

1.2
a) If the calendar time is stored as a signed 32-bit integer, in what year will it overflow. (Unix calendar time overflow)

signed int = $2^{32-1} – 1 = 2^{31} - 1 = (2^{10} * 2^{10} * 2^{10} * 2^{1}) – 1$ = (1024 * 1024 * 1024 * 2) – 1 = 2147483648 – 1 = 2147483647
Seconds per day = 24 * 60 * 60 = 86400 sec
Seconds per year = 365 * 86400 = 31536000 sec
2147483647 / 31536000 = 68.096259735 years
Unix calendar time starts from 1970
1970 + 68.096258735 = 2038.096259735
Or, using date command

```
date -d @2147483647
Mon Jan 18 19:14:07 PST 2038
```

b) if the process time is stored as a signed 32-bit integer, and if the system counts 100 ticks per second, after how many days will the value overflow?

signed int = $2^{32-1} – 1 = 2^{31} - 1 = (2^{10} * 2^{10} * 2^{10} * 2^{1}) – 1$ = (1024 * 1024 * 1024 * 2) – 1 = 2147483648 – 1 = 2147483647
2147483647 / 100 = 21474836.47
Seconds per day = 24 * 60 * 60 = 86400 sec
Process time overflow = 21474836.47 / 86400 = 248.551348032 days

1.3

```
#include <stdio.h>
int main(void)
{
    char c;
    while ((c=getchar()) != EOF)
        putchar(c);
}
```

- the return type of getchar() is integer.
- EOF is defined -1
- If system uses signed char as a default, it will be worked.
- If system uses unsigned char, then the while loop will not be finished because the return value will be unsigned character, and EOF is -1. So, comparing these two values never matched.

1.4
a)

```
#include <stdio.h>
#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <fcntl.h>

int main()
{
	int fd;
	int i;

	for (i = 0 ; (fd = open("hw01_1_4a.c", O_RDONLY)) != EOF; i++) ;

	printf("Max open = %d\n", i);
}
```

```
Max open = 1021
```

b)

```
#include <unistd.h>
#include <limits.h>
#include <stdio.h>

int main()
{
	int max_value;

	max_value = sysconf(_SC_OPEN_MAX);
	printf("SC_OPEN_MAX = %d\n", max_value);
}
```

```
SC_OPEN_MAX = 1024
```

1.5

a)

```
getconf -a | grep CLK_TCK
CLK_TCK                            100
```

b)

```
#include <unistd.h>
#include <limits.h>
#include <stdio.h>

int main()
{
	int clock_tck;

	clock_tck = sysconf(_SC_CLK_TCK);
	printf("Clock Ticks = %d\n", clock_tck);
}
```

```
Clock Ticks = 100
```

1.6

```
/* hw01 - 1.6 */

#include <stdio.h>

void fn(char *str)
{
	if (*str) {
```

```
            fn(str+6);
            putchar(*str-1);
        }
        return;
}

int main(void)
{
        /* Some, World Currencies */
        char *str = "\"YeN! cEnT! pEsO! KrOnEr! pEnCe! LeMpIrA! pUlS! Hi!!\"";

        fn(str);
        putchar('\n');

        return;
}
```

a) Good Job!

b)

str

“	Y	e	N	!		c	E	n	T	!		p	E	s	O	!		K	r	O	n	E	r	!		p	E	n	C	e	!		L	e	M	p	I	r	A	!		p	U	l	S	!		H	i	!	!	“	\0
34						39						112						75						33						101						112						112						72					
33						38						111						74						32						100						111						111						71					
!						**b**						**o**						**J**												**d**						**o**						**o**						**G**					

str is moved by 6 recursively, and reached to the end of string.
And, if nothing at the point of str, then return to previous and display a character. If the character code is 72, then print the character of ascii code 71.
And, repeat it until reached to the beginning of string.

c)
str is the start address of string.
It is initialized with string, and its location will be code address space.

1.7

```
int i;main(){for(;i["]<i;++i){--i;}"];read('-'-'-',i+++"hell\
o, world!\n",'/'/'/'));}read(j,i,p){write(j/p+p,i---j,i/i);}
```

a) hello, world!

b)

```
int i;

main()
{
        for ( ; i["]<i;++i){--i;}"]; read('-'-'-', (i++) + "hello, world!\n", '/'/'/'));
}

read(j, i, p)
{
        write(j/p + p, ((i--) - j), i/i);
}

i=0, ‘h’
i=1, ‘e’
i=2, ‘l’
```

```
i=3, ‘l’
i=4, ‘o’
i=5, ‘,’
i=6, ‘ ’
i=7, ‘w’
i=8, ‘o’
i=9, ‘r’
i=10, ‘l’
i=11, ‘d’
i=12, ‘!’
i=13, ‘\0’
```

```
Here,
j is ‘-’-’-’
i is i++ + “hello, world!\n”
p is ‘/’/’/’
```

```
write() function is
ssize_t write(int fd, const void *buf, size_t count);
```

```
so, int fd = j/p + p = 1
size_t count = i/i = 1
And,  ((i--) - j) is pointed to buffer stored a character.
```

c)

```
int i is global and uninitialized variable, and located in data memory area (BSS).
```