

실습 7주차

CPS LAB

프로세스 정보

목차

1. 예제풀이

2. 연습문제

예제 풀이

예제 6-1

sysinfo()함수로 메모리 크기 검색하기

```
SYSINFO(2)                                Linux Programmer's Manual

NAME
  sysinfo - return system information

SYNOPSIS
  #include <sys/sysinfo.h>

  int sysinfo(struct sysinfo *info);

DESCRIPTION
  sysinfo() returns certain statistics on memory and swap usage, as well as the load average.

  Until Linux 2.3.16, sysinfo() returned information in the following structure:

      struct sysinfo {
          long uptime;           /* Seconds since boot */
          unsigned long loads[3]; /* 1, 5, and 15 minute load averages */
          unsigned long totalram; /* Total usable main memory size */
          unsigned long freeram;  /* Available memory size */
          unsigned long sharedram; /* Amount of shared memory */
          unsigned long bufferram; /* Memory used by buffers */
          unsigned long totalswap; /* Total swap space size */
          unsigned long freeswap; /* Swap space still available */
          unsigned short procs;   /* Number of current processes */
          char _f[22];            /* Pads structure to 64 bytes */
      };

RETURN VALUE
  On success, sysinfo() returns zero. On error, -1 is returned, and errno is set to indicate the cause of the error.

NOTES
  All of the information provided by this system call is also available via /proc/meminfo and /proc/loadavg.
```

예제 풀이

예제 6-1

sysinfo()함수로 메모리 크기 검색하기

- top 명령어를 통해 각 프로세스, 및 평균 부하 정보를 얻을 수 있음

```
top - 17:01:55 up 2 days, 8 min, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 290 total, 2 running, 287 sleeping, 1 stopped, 0 zombie
%Cpu(s):  0.2/0.1  0[
MiB Mem : 10.5/31836.2  [
MiB Swap:  0.0/2048.0  [

  PID USER      PR  NI    VIRT    RES    SHR  S  %CPU  %MEM     TIME+ COMMAND
 2096 kyhooon   20   0 4459184 318468 115292 R   3.7   1.0 261:01.03 gnome-shell
 1866 kyhooon    9 -11 3511884 23840  18308 S   0.7   0.1  18:51.96 pulseaudio
 1379 root      20    0 1953892 20304  13684 S   0.3   0.1   3:48.93 teamviewerd
 5637 kyhooon   20    0 1055076 65256  47248 S   0.3   0.2   1:17.39 gnome-terminal-
16647 kyhooon   20    0   16512   4548   3616 R   0.3   0.0   0:00.39 top
   1 root      20    0  169716  12996   8220 S   0.0   0.0   0:04.19 systemd
   2 root      20    0         0         0         0 S   0.0   0.0   0:00.02 kthreadd
   3 root        0 -20         0         0         0 I   0.0   0.0   0:00.00 rcu_gp
   4 root        0 -20         0         0         0 I   0.0   0.0   0:00.00 rcu_par_gp
   5 root        0 -20         0         0         0 I   0.0   0.0   0:00.00 slub_flushwq
   6 root        0 -20         0         0         0 I   0.0   0.0   0:00.00 netns
   8 root        0 -20         0         0         0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
```

예제 풀이

예제 6-1

sysinfo()함수로 메모리 크기 검색하기

```
1 #include <sys/sysinfo.h>
2 #include <stdio.h>
3
4 int main(void) {
5     struct sysinfo info;
6     sysinfo(&info);
7
8     printf("사용가능한 총 메모리 크기: %ld\n", info.totalram);
9
10    printf("사용가능한 메모리의 크기: %ld\n", info.freeram);
11
12    printf("현재 실행 중인 프로세스 수: %d\n", info.procs);
13
14    return 0;
15 }
16
17 }
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch6$ vim ch6_1.c
kyhooon@kyh:~/sysprogram_practice/p_ch6$
kyhooon@kyh:~/sysprogram_practice/p_ch6$ gcc ch6_1.c -o ch6_1.out
kyhooon@kyh:~/sysprogram_practice/p_ch6$
kyhooon@kyh:~/sysprogram_practice/p_ch6$ ./ch6_1.out
사용가능한 총 메모리 크기: 33382658048
사용가능한 메모리의 크기: 27111538688 바이트
현재 실행 중인 프로세스 수: 1025
kyhooon@kyh:~/sysprogram_practice/p_ch6$
```

예제 풀이

예제 6-2

getpid(), getppid() 함수로 PID, PPID 검색하기

```
GETPID(2)                                     Linux Programmer's Manual                                     GETPID(2)

NAME
    getpid, getppid - get process identification

SYNOPSIS
    #include <sys/types.h>
    #include <unistd.h>

    pid_t getpid(void);
    pid_t getppid(void);

DESCRIPTION
    getpid() returns the process ID (PID) of the calling process. (This is often used by routines that generate unique temporary filenames.)

    getppid() returns the process ID of the parent of the calling process. This will be either the ID of the process that created this process using fork(), or, if that process has already terminated, the ID of the process to which this process has been reparented (either init(1) or a "subreaper" process defined via the prctl(2) PR_SET_CHILD_SUBREAPER operation).

ERRORS
    These functions are always successful.
```

예제 풀이

예제 6-2

getpid(), getppid() 함수로 PID, PPID 검색하기

```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main(void) {
5
6     printf("PID : %d\n", (int)getpid());
7
8     printf("PPID : %d\n", (int)getppid());
9
10    return 0;
11 }
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch6$ ./ch6_2.out
PID : 31506
PPID : 5648
kyhooon@kyh:~/sysprogram_practice/p_ch6$ ps
  PID TTY          TIME CMD
 5648 pts/0    00:00:00 bash
 31507 pts/0    00:00:00 ps
kyhooon@kyh:~/sysprogram_practice/p_ch6$
```


예제 풀이

예제 6-3

getpgrp(), getpgid() 함수로 PGID 검색하기

SYNOPSIS

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

int setpgid(pid_t pid, pid_t pgid);
pid_t getpgid(pid_t pid);

pid_t getpgrp(void);           /* POSIX.1 version */
pid_t getpgrp(pid_t pid);       /* BSD version */
```

All of these interfaces are available on Linux, and are used for getting and setting the process group ID (PGID) of a process. The preferred, POSIX.1-specified ways of doing this are: **getpgrp(void)**, for retrieving the calling process's PGID; and **setpgid()**, for setting a process's PGID.

getpgid() returns the PGID of the process specified by pid. If pid is zero, the process ID of the calling process is used. (Retrieving the PGID of a process other than the caller is rarely necessary, and the POSIX.1 **getpgrp()** is preferred for that task.)

예제 풀이

예제 6-3

getpgrp(), getpgid() 함수로 PGID 검색하기

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/types.h>
4
5 int main(void) {
6
7     printf("PID : %d\n", (int)getpid());
8
9     printf("PGRP : %d\n", (int)getpgrp());
10
11     printf("PGID(0) : %d\n", (int)getpgid(0));
12
13     return 0;
14 }
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch6$ vim ch6_3.c
kyhooon@kyh:~/sysprogram_practice/p_ch6$
kyhooon@kyh:~/sysprogram_practice/p_ch6$ gcc ch6_3.c -o ch6_3.out
kyhooon@kyh:~/sysprogram_practice/p_ch6$
kyhooon@kyh:~/sysprogram_practice/p_ch6$ ./ch6_3.out
PID : 31793
PGRP : 31793
PGID(0) : 31793
kyhooon@kyh:~/sysprogram_practice/p_ch6$
```

예제 풀이

예제 6-4

getsid()함수로 세션 ID 검색하기

```
GETSID(2)                                     Linux Programmer's Manual                                     GETSID(2)

NAME
    getsid - get session ID

SYNOPSIS
    #include <sys/types.h>
    #include <unistd.h>

    pid_t getsid(pid_t pid);

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

    getsid():
        _XOPEN_SOURCE >= 500
        || /* Since glibc 2.12: */ _POSIX_C_SOURCE >= 200809L

DESCRIPTION
    getsid(0) returns the session ID of the calling process. getsid() returns the session ID of the process with process ID pid.
    If pid is 0, getsid() returns the session ID of the calling process.

RETURN VALUE
    On success, a session ID is returned. On error, (pid_t) -1 will be returned, and errno is set appropriately.
```

예제 풀이

예제 6-4

getsid()함수로 세션 ID 검색하기

```
1 #include <unistd.h>
2 #include <stdio.h>
3
4 int main(void) {
5
6     printf("PID : %d\n", (int)getpid());
7
8     printf("PGID : %d\n", (int)getpgrp());
9
10    printf("SID : %d\n", (int)getsid(0));
11
12    return 0;
13 }
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch6$ ./ch6_4.out
```

```
PID : 32012
```

```
PGID : 32012
```

```
SID : 5648
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch6$ ps
```

PID	TTY	TIME	CMD
-----	-----	------	-----

<u>5648</u>	pts/0	00:00:00	bash
-------------	-------	----------	------

32013	pts/0	00:00:00	ps
-------	-------	----------	----

```
kyhooon@kyh:~/sysprogram_practice/p_ch6$
```

예제 풀이

예제 6-5

times() 함수로 실행 시간 측정하기

TIMES(2)

NAME

times - get process times

SYNOPSIS

#include <sys/times.h>

clock_t times(struct tms *buf);

times() stores the current process times in the struct tms that buf points to. The struct tms is as defined in <sys/times.h>:

```
struct tms {
    clock_t tms_utime; /* user time */
    clock_t tms_stime; /* system time */
    clock_t tms_cutime; /* user time of children */
    clock_t tms_cstime; /* system time of children */
};
```

The tms_utime field contains the CPU time spent executing instructions of the calling process. The tms_stime field contains the CPU time spent executing inside the kernel while performing tasks on behalf of the calling process.

The tms_cutime field contains the sum of the tms_utime and tms_cutime values for all waited-for terminated children. The tms_cstime field contains the sum of the tms_stime and tms_cstime values for all waited-for terminated children.

RETURN VALUE

times() returns the number of clock ticks that have elapsed since an arbitrary point in the past. The return value may overflow the possible range of type clock_t. On error, (clock_t) -1 is returned, and errno is set appropriately.



예제 풀이

예제 6-5

times() 함수로 실행 시간 측정하기

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <sys/times.h>
4 #include <time.h>
5 #include <unistd.h>
6
7 int main(void) {
8
9     int i;
10    struct tms tmsbuf;
11    clock_t ct, t1, t2;
12
13    ct = sysconf(_SC_CLK_TCK);
14    printf("클록 틱 값 : %ld\n", ct);
15
16    if( (t1 = times(&tmsbuf)) == -1 ) {
17        perror("times");
18        exit(1);
19    }
20
21    for(i = 0; i < 99999999; i++);
22
23    if( (t2 = times(&tmsbuf)) == -1 ) {
24        perror("times");
25        exit(1);
26    }
27
28    printf("t1 : %ld\n", t1);
29    printf("t2 : %ld\n", t2);
30    printf("프로세스 사용한 사용자 모드 실행 시간 : %ld\n", tmsbuf.tms_utime);
31    printf("프로세스 사용한 커널 모드 실행 시간 : %ld\n", tmsbuf.tms_stime);
32
33    printf("실 사용한 시간: %.2f sec\n", (double) (t2 - t1) / ct );
34    printf("사용자 모드 시간: %.2f sec\n", (double) tmsbuf.tms_utime / ct );
35    printf("커널 모드 시간: %.2f sec\n", (double) tmsbuf.tms_stime / ct );
36
37    return 0;
38 }
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch6$ ./ch6_5.out
클록 틱 값 : 100
t1 : 1742157679
t2 : 1742157692
프로세스 사용한 사용자 모드 실행 시간 : 12
프로세스 사용한 커널 모드 실행 시간 : 0
실 사용한 시간: 0.13 sec
사용자 모드 시간: 0.12 sec
커널 모드 시간: 0.00 sec
kyhooon@kyh:~/sysprogram_practice/p_ch6$
```

예제 풀이

예제 6-6

environ 전역 변수로 환경변수 검색하기

```
kyhooon@kyh:~/sysprogram_practice/p_ch6$ env  
SHELL=/bin/bash
```

```
CONDA_EXE=/home/kyhooon/anaconda3/bin/conda
```

```
JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
```

```
PWD=/home/kyhooon/sysprogram_practice/p_ch6
```

```
HOME=/home/kyhooon
```

예제 풀이

예제 6-6

ENVIRON(7)

Linux Programmer's Manual

ENVIRON(7)

NAME

environ - user environment

SYNOPSIS

extern char **environ;

DESCRIPTION

The variable environ points to an array of pointers to strings called the "environment". The last pointer in this array has the value NULL. (This variable must be declared in the user program, but is declared in the header file <unistd.h> if the _GNU_SOURCE feature test macro is defined.) This array of strings is made available to the process by the exec(3) call that started the process. When a child process is created via fork(2), it inherits a copy of its parent's environment.

By convention the strings in environ have the form "name=value". Common examples are:

USER The name of the logged-in user (used by some BSD-derived programs).

LOGNAME

The name of the logged-in user (used by some System-V derived programs).

HOME

A user's login directory, set by **login(1)** from the password file **passwd(5)**.

LANG

The name of a locale to use for locale categories when not overridden by **LC_ALL** or more specific environment variables such as **LC_COLLATE**, **LC_CTYPE**, **LC_MESSAGES**, **LC_MONETARY**, **LC_NUMERIC**, and **LC_TIME** (see **locale(7)** for further details of the **LC_*** environment variables).

PATH

The sequence of directory prefixes that **sh(1)** and many other programs apply in searching for a file known by an incomplete pathname. The prefixes are separated by ':'. (Similarly one has **CDPATH** used by some shells to find the target of a change directory command, **MANPATH** used by **man(1)** to find manual pages, and so on)

예제 풀이

예제 6-6

environ 전역 변수로 환경변수 검색하기

```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 extern char **environ;
5
6 int main(void) {
7
8     char **env;
9
10    env = environ;
11
12    while(*env) {
13        printf("%s\n", *env);
14        env++;
15    }
16
17    return 0;
18 }
```

예제 풀이

예제 6-7

main() 함수 인자로 환경변수 검색하기

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[], char **envp) {
4
5     char **env = envp;
6
7     while(*env) {
8         printf("%s\n", *env);
9         env++;
10    }
11 }
```

예제 풀이

예제 6-8

getenv() 함수로 환경변수 검색하기

```
GETENV(3)                                     Linux Programmer's Manual                                     GETENV(3)

NAME
    getenv, secure_getenv - get an environment variable

SYNOPSIS
    #include <stdlib.h>

    char *getenv(const char *name);

    char *secure_getenv(const char *name);

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

    secure_getenv(): _GNU_SOURCE

DESCRIPTION
    The getenv() function searches the environment list to find the environment variable name, and returns a pointer to the corresponding value string.
```

예제 풀이

예제 6-8

getenv() 함수로 환경변수 검색하기

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void) {
5
6     char *val;
7
8     val = getenv("PATH");
9     if( val == NULL)
10         printf("PATH is not defined\n");
11     else
12         printf("PATH = %s\n", val);
13
14     return 0;
15 }
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch6$ ./ch6_8.out
PATH = /home/kyhooon/anaconda3/condabin:/usr/local/sbin:/usr/local/bin:
in:$JAVA_HOME
kyhooon@kyh:~/sysprogram_practice/p_ch6$
```

예제 풀이

예제 6-9

putenv() 함수로 환경변수 검색하기

```
PUTENV(3) Linux Programmer's Manual PUTENV(3)

NAME
    putenv - change or add an environment variable

SYNOPSIS
    #include <stdlib.h>

    int putenv(char *string);

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

    putenv(): _XOPEN_SOURCE
               || /* Glibc since 2.19: */ _DEFAULT_SOURCE
               || /* Glibc versions <= 2.19: */ _SVID_SOURCE

DESCRIPTION
    The putenv() function adds or changes the value of environment variables. The argument string is of the form name=value. If name does not already exist in the environment, then string is added to the environment. If name does exist, then the value of name in the environment is changed to value. The string pointed to by string becomes part of the environment, so altering the string changes the environment.

RETURN VALUE
    The putenv() function returns zero on success, or nonzero if an error occurs. In the event of an error, errno is set to indicate the cause.
```

예제 풀이

예제 6-9

putenv() 함수로 환경변수 검색하기

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void) {
5
6     char *val;
7
8     val = getenv("TERM");
9     if( val == NULL )
10         printf("TERM is not defined\n");
11     else
12         printf("1. TERM=%s\n", val);
13
14     putenv("TERM=tv100");
15
16     val = getenv("TERM");
17     printf("2. TERM=%s\n", val);
18
19     return 0;
20 }
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch6$ ./ch6_9.out
1. TERM=xterm-256color
2. TERM=tv100
kyhooon@kyh:~/sysprogram_practice/p_ch6$
```


예제 풀이

예제 6-10

setenv() 함수로 환경변수 설정하기

```
SETENV(3) Linux Programmer's Manual

NAME
    setenv - change or add an environment variable

SYNOPSIS
    #include <stdlib.h>

    int setenv(const char *name, const char *value, int overwrite);

    int unsetenv(const char *name);
```

DESCRIPTION

The **setenv()** function adds the variable name to the environment with the value value, if name does not already exist. If name does exist in the environment, then its value is changed to value if overwrite is nonzero; if overwrite is zero, then the value of name is not changed (and **setenv()** returns a success status). This function makes copies of the strings pointed to by name and value (by contrast with **putenv(3)**).

The **unsetenv()** function deletes the variable name from the environment. If name does not exist in the environment, then the function succeeds, and the environment is unchanged.

RETURN VALUE

The **setenv()** function returns zero on success, or -1 on error, with errno set to indicate the cause of the error.

The **unsetenv()** function returns zero on success, or -1 on error, with errno set to indicate the cause of the error.

예제 풀이

예제 6-10

setenv() 함수로 환경변수 설정하기

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void) {
5
6     char *val;
7
8     val = getenv("TERM");
9     if( val == NULL )
10         printf("TERM is not defined\n");
11     else
12         printf("1. TERM=%s\n", val);
13
14     setenv("TERM", "vt100", 0);
15     val = getenv("TERM");
16     printf("2. TERM=%s\n", val);
17
18     setenv("TERM", "vt100", 1);
19     val=getenv("TERM");
20     printf("3. TERM=%s\n", val);
21
22     return 0;
23 }
```

```
kyhooon@kyh:~/sysprogram_practice/p_ch6$ ./ch6_10.out
1. TERM=xterm-256color
2. TERM=xterm-256color
3. TERM=vt100
kyhooon@kyh:~/sysprogram_practice/p_ch6$
```

연습 문제



감사합니다.

CPS LAB