システムログラミング II 令和元年度 前期末試験

(2019.08.09 重村 哲至)　　IE4　　＿＿番　氏名＿＿模範解答＿＿

1　プロセスとシグナル

以下の問に答えなさい．なおプログラムを書く場合は，`#include` ディレクティブは省略して良い．

1. ターミナルでプログラムを起動した後，プログラムが終了する前に，Ctrl-C を入力した場合と，Ctrl-Z を入力した場合で，一般に何が異なるか答えなさい．(5 点)

   Ctrl-C を入力した場合はプログラムが終了する．Ctrl-Z を入力した場合はプログラムは一時的に停止するだけで終了していない．

2. バグのあるプログラムが暴走してしまった場合，前の問題のどちらの操作が適切か，理由も含めて答えなさい．(5 点)

   Ctrl-C が適切である．Ctrl-Z ではプログラムが終了していないのでプロセスが残っている．プロセスがメモリなどの資源を解放しないので無駄になる．

3. 付録の C 言語プログラム p1.c の実行結果を答えなさい．なお，表示されるのが実行開始から何秒後になるかも，同じ行の右側に書きなさい．(5 点)

実行結果	時刻 (秒)
$ p1	
!!!	1
!!!	2
!!!	3
$	

4. 有効な `signal()` を用い，フラグを用いない方法で p1.c と同じ実行結果になるプログラム p2.c を書きなさい．(6 点)

```
// p2.c
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
void handler(int n) {
}
int main() {
  signal(SIGALRM, handler);
  for (int i=0; i<3; i++) {
    alarm(1);
    pause();
    printf("!!!\n");
  }
}
```

5. `signal()` も，フラグも用いない方法で p1.c と同じ実行結果になるプログラム p3.c を書きなさい．(6 点)

```
// p3.c
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
int main() {
  for (int i=0; i<3; i++) {
    sleep(1);
    printf("!!!\n");
  }
}
```

システムログラミング II 令和元年度 前期末試験

(2019.08.09 重村 哲至)　　IE4　　＿＿＿番　氏名＿＿＿＿＿＿模範解答＿＿＿＿＿＿

2　環境変数

以下の問に答えなさい．

1. 次の実行結果の空欄①〜③に表示されるものを答えなさい．(3 点 ×4 問=12 点)

```
$ export A=B
$ printenv A
①
$ env A=C printenv A
②
$ printenv A
③
$ echo $A $A
④
```

空欄	表示されるもの
①	B
②	C
③	B
④	B B

2. 付録の C 言語プログラム p4.c を以下の順で実行しました．3〜5 行の実行結果（表示）を答えなさい．なお，何も表示されない場合は解答欄に「×」を記入しなさい．(3 点 ×3 問=9 点)

```
1 $ export A=abc
2 $ export B=123
3 $ ./p4 A - printenv A
4 $ ./p4 A - printenv B
5 $ ./p4 A B - printenv B
```

行	表示されるもの
3	×
4	123
5	×

3. 付録の C 言語プログラム p4.c を，授業演習で使用したコンピュータで実行した結果が次のようになりました．

```
1 $ ./p4 - date
2 2019 年 8 月 8 日 木曜日 09 時 39 分 17 秒 JST
3 $ ./p4 LC_TIME - date
4 Thu Aug  8 09:39:09 JST 2019
5 $ ./p4 LC_TIME - env TZ=Singapore date
6 Thu Aug  8 08:44:05 +08 2019
```

(a) このことから LC_TIME 環境変数の値が何であったか答えなさい．(3 点)

ja_JP.UTF-8

(b) 5 行の date コマンド実行時の LC_TIME，TZ 環境変数の状態を答えなさい．(4 点)

LC_TIME 環境変数は存在しない．
TZ 環境変数の値は Singapore になっている．

4. p4.c は何をするプログラムか簡潔に答えなさい．(5 点)

'-' より前のコマンド行引数で示される環境変数を削除した後，'-' より後ろのコマンド行引数で表現されるコマンドを実行するプログラムである．

システムログラミングII 令和元年度 前期末試験

(2019.08.09 重村 哲至)　　IE4　　＿＿＿番　氏名＿＿＿模範解答＿＿＿

3　プロセス生成とプログラム実行

1. 付録の C 言語プログラム p5.c の実行結果が次のようになる時，プログラム中の空欄①，②に適切な記述を答えなさい．(5 点 ×2 問=10 点)

```
$ ./p5
A=1
B=2
```

① `"printenv", NULL`

② `"A=1", "B=2", NULL`

2. 付録の C 言語プログラム p6.c の実行結果が次のようになる時，プログラム中の空欄③に適切な記述を答えなさい．なお，②の記述は p5.c と同じものとします．(5 点)

```
$ ./p6
2
```

③ `"printenv", "B", NULL`

3. 付録の C 言語プログラム p7.c の実行結果を答えなさい．(5 点)

```
$ ./p7
```

子：a=20
親：a=10

4. 付録の C 言語プログラム p7.c の実行結果で表示される値が前問のようになる理由を簡単に説明しなさい．(5 点)

変数 a は fork すると親プロセスと子プロセスで別々のインスタンスを持つようになる．子プロセスが a の値を 20 に変更しても，親プロセスの変数 a の値は変化しない．そのため，親プロセスの表示は初期状態の 10 になる．

4　シェルの改造

付録の簡易版シェル myshell.c が，次の実行例のようにカレントディレクトリのパスを格納した PWD 環境変数を持ち，管理するように改造します．

ヒント：getecwd() 関数を用いると良い．getecwd() 関数のマニュアルの要約を付録に掲載する．

```
1  $ ./myshell
2  Command: printenv PWD
3  Command: cd /Users/sigemura
4  Command: printenv PWD
5  /Users/sigemura
6  Command: ls
7  Desktop
8  Documents
9  Downloads
10 ... 省略 ...
11 Command: cd Downloads
12 Command: printenv PWD
13 /Users/sigemura/Downloads
14 Command: cd ..
15 Command: printenv PWD
16 /Users/sigemura
17 Command:
```

システムログラミング II 令和元年度 前期末試験

(2019.08.09 重村 哲至)　　IE4　　＿＿＿番　氏名＿＿＿＿模範解答＿＿＿＿

実行例の解説

2 行 myshell の起動時には，PWD 環境変数は存在しないか，または，親プロセスが同じ名前の環境変数を持っていれば，それの値を引き継いだものとします．

3 行 cd コマンドを実行すると PWD 環境変数の値が変化します．

1. PWD 環境変数の管理は，シェル自身のプロセス，外部コマンドを実行するプロセスのどちらで実行すべきか答えなさい．(5 点)

   シェル自身のプロセス

2. myshell.c の行番号で改造箇所を明示し，改造後のプログラムを書きなさい．(10 点)

   ３１ 行 ～　３１　行

```
} else {
  char *buf=getcwd(NULL, 0);
  setenv("PWD", buf, 1);
  free(buf);
}
```

付録

```
// p1.c
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
volatile sig_atomic_t flg;
void handler(int n) {
  flg = 1;
}
int main() {
  signal(SIGALRM, handler);
  for (int i=0; i<3; i++) {
    flg = 0;
    alarm(1);
    while (flg==0) {
    }
    printf("!!!\n");
  }
  return 0;
}
```

```
// p4.c
#include <stdio.h>
#include <stdlib.h> // unsetenv
#include <unistd.h> // exec
#include <string.h> // strcmp
extern char **environ;
int main(int argc, char *argv[]) {
  int i;
  for (i=1; argv[i]!=NULL; i++) {
    if (strcmp(argv[i],"-")==0) {
      i++;          // argv[i] が "-" なら
      break;        //  ここが実行される
    }
    unsetenv(argv[i]);
  }
  if (argv[i]!=NULL) {
    execvp(argv[i], &argv[i]);
```

システムログラミングII 令和元年度 前期末試験

(2019.08.09 重村 哲至)　　IE4　　＿＿＿番　氏名＿＿＿＿＿＿模範解答＿＿＿＿＿＿

```
    perror(argv[i]);
    return 1;
  }
  return 0;
}
```

```
// p5.c
#include <stdio.h>
#include <unistd.h>
char *args[] = { ① };
char *envs[] = { ② };
int main() {
  execve("/usr/bin/printenv", args, envs);
  perror("printenv");
  return 1;
}
```

```
// p6.c
#include <stdio.h>
#include <unistd.h>
char *args[] = { ③ };
char *envs[] = { ② };
int main() {
  execve("/usr/bin/printenv", args, envs);
  perror("printenv");
  return 1;
}
```

```
// p7.c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
int main(int argc, char* argv[]) {
  int a=10;
  int pid=fork();
  if (pid==0) {
    a=20;
    printf("子：a=%d\n", a);
  } else {
    int stat;
    wait(&stat);
    printf("親：a=%d\n", a);
  }
  return 0;
}
```

```
1  // myshell.c
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <unistd.h>
6  #include <sys/wait.h>
7  #include <ctype.h>
8  #define MAXLINE 1000
9  #define MAXARGS 60
10
11 int parse(char *p, char *args[]) {
12   int i=0;
13   for (;;) {
14     while (isspace(*p))
15       *p++ = '\0';
16     if (*p=='\0' || i>=MAXARGS) break;
17     args[i++] = p;
18     while (*p!='\0' && !isspace(*p))
19       p++;
20   }
21   args[i] = NULL;
22   return *p=='\0';
23 }
24
25 void execute(char *args[]) {
26   if (strcmp(args[0], "cd")==0) {
27     if (args[1]==NULL) {
28       fprintf(stderr,"cd の引数が不足\n");
29     } else if (chdir(args[1])<0) {
30       perror(args[1]);
31     }
32   } else {
33     int pid, status;
34     if ((pid = fork()) < 0) {
```

システムログラミング II 令和元年度 前期末試験

(2019.08.09 重村 哲至)　　IE4　　＿＿＿番　氏名＿＿＿＿＿＿模範解答＿＿＿＿＿＿

```
35        perror("fork");
36        exit(1);
37      }
38      if (pid==0) {
39        int i;
40        for (i=0; args[i]!=NULL; i++) {
41          if (putenv(args[i])!=0) break;
42        }
43        if (args[i]!=NULL) {
44          execvp(args[i], &args[i]);
45          perror(args[i]);
46        }
47        exit(1);
48      }
49      while (wait(&status) != pid)
50        ;
51    }
52  }
53
54  int main() {
55    char buf[MAXLINE+2];
56    char *args[MAXARGS+1];
57    for (;;) {
58      printf("Command: ");
59      if (fgets(buf,MAXLINE+2,stdin)==NULL) {
60        printf("\n");
61        break;
62      }
63      if (strchr(buf, '\n')==NULL) {
64        fprintf(stderr, "行が長すぎる\n");
65        return 1;
66      }
67      if (!parse(buf,args)) {
68        fprintf(stderr, "引数が多すぎる\n");
69        continue;
70      }
71      if (args[0]!=NULL) execute(args);
72    }
73    return 0;
74  }
```

getcwd() 関数のマニュアル要約

書式： `#include <unistd.h>`
`char *getcwd(char *buf, int size);`

説明：
getcwd() 関数はカレントディレクトリの絶対パスを buf によって示されるメモリ領域に書き込みます．getcwd() 関数は buf のポインターを返します．size は buf が示すメモリ領域のバイト単位のサイズです．もしも buf が NULL なら，getcwd() 関数が必要なメモリ領域を malloc() を用いて自動的に割り当てます．この場合は size は無視されます．malloc() で割り当てられた領域は，後で free() を用いて解放する必要があります．

使用例：

```
char *path=getcwd(NULL, 0);
printf("pwd=%s\n", path);
free(path);
```