システムログラミング II 令和 4 年度 前期末試験

(2022.08.04 重村 哲至)　　IE4　　＿＿＿番　氏名＿＿＿＿＿＿模範解答＿＿＿＿＿＿

付録 4 にヒントがあるので適宜参照すること．

1　語句に関する問題

空欄に最適な言葉を語群から記号で答えなさい．
(1 点 ×15 問=15 点)

環境変数はシェルが管理し，プログラム起動時にシェルからプログラムに渡される．プログラムは渡された環境変数を C 言語の **(1)** 関数や **(2)** グローバル変数を使用して参照することができる．例えば date コマンド（プログラム）は，LC_TIME 環境変数の値により表示に用いる **(3)** を，TZ 環境変数の値によりにより **(4)** を判断する．環境変数の一覧や値を知るためには **(5)** コマンド（プログラム）を用いる．

新しいプログラムを実行する方式は，プロセスの生成とプログラムの実行を，1 つのシステムコールで行う **(6)** 方式と，2 つのシステムコールで行う **(7)** 方式がある．**(7)** 方式では，**(8)** システムコールで子プロセスを生成し，その後，子プロセスが自ら **(9)** 処理を行った上で **(10)** システムコールで新しいプログラムを実行する．**(9)** 処理として標準入出力を close して open し直すことで **(11)** が実現できる．**(12)** プログラムは，自分の環境変数を変更した上で目的のプログラムを実行するプログラムである．

UNIX シェルは **(13)** 方式のコマンドインタプリタである．**(14)** コマンドはシェルの子プロセスが実行する．**(15)** コマンドはシェル自身が実行する．

語群：(あ)CLI(Command Line Interface)，(い)env，(う)environ，(え)execve，(お)fork，(か)fork–exec，(き)getenv，(く)printenv，(け)spawn，(こ) タイムゾーン，(さ) リダイレクト，(し) 外部，(す) 言語，(せ) 初期化，(そ) 内部

(1)	(き)	(2)	(う)	(3)	(す)
(4)	(こ)	(5)	(く)	(6)	(け)
(7)	(か)	(8)	(お)	(9)	(せ)
(10)	(え)	(11)	(さ)	(12)	(い)
(13)	(あ)	(14)	(し)	(15)	(そ)

2　内部コマンドと外部コマンド

シェルの外部コマンドとして実現可能なものに○，実現不可能なものに × を付けなさい．(2 点 ×7 問=14 点)

現在時刻の表示	○
カレントディレクトリの表示	○
カレントディレクトリの変更	×
環境変数の表示	○
環境変数の作成・変更	×
環境変数の削除	×
ファイルのコピー	○

3　環境変数の役割

次の実行例は Linux や macOS の標準的なシェルを使用した場合のものです．実行例の空欄（##(?)##）に適切な入力や表示を答えなさい．(5 点 ×3 問=15 点)

```
$ ./mycp a.txt b.txt            # 実行できる
$ mycp b.txt c.txt              # 実行できない
bash: mycp: command not found
$ printenv PATH
/bin:/usr/bin
$ PATH=##(A)##
$ mycp b.txt c.txt              # 実行できる
$ printenv PATH
##(B)##
$ date
2022年 8月 1日 月曜日 23時39分27秒 JST
$ printenv LC_TIME
$ ##(C)##                       # ロケールをCにする
$ date
Mon Aug  1 23:39:44 JST 2022
```

(A)	$PATH:.
(B)	/bin:/usr/bin:.
(C)	export LC_TIME=C

システムログラミング II 令和 4 年度 前期末試験

(2022.08.04 重村 哲至)　　IE4　　＿＿＿番　氏名＿＿＿＿＿＿模範解答＿＿＿＿＿＿

4　実行結果

1. 付録 1 の p0.c の出力を答えなさい．(5 点)

```
X=Y
A=B
B=C
```

2. 付録 1 の p1.c の出力を答えなさい．(5 点)

```
B
```

3. 付録 1 の p2.c の出力を答えなさい．(5 点)

```
child
parent
exit
```

4. 付録 1 の p3.c の出力を答えなさい．(5 点)

```
child
parent
```

5. 付録 1 の p4.c の出力を答えなさい．(5 点)

```
child
exit
parent
exit
```

6. 付録 1 の p5.c の出力を答えなさい．(5 点)

```
child
echo
parent
```

5　printenv プログラム

付録 2 の C 言語プログラムは，printenv コマンドのクローン myprintenv プログラムのソースです．プログラム中の空欄（##(?)##）に適切な記述を答えなさい．(4 点 ×4 問=16 点)

(A) `argc==1`

(B) `environ[i]!=NULL`

(C) `getenv(argv[1])`

(D) `"%s\n", env`

6　system 関数

付録 3 の C 言語プログラムは，system 関数のクローン mysystem 関数のソースです．8 行の空欄 (##(A)##) には「return 127」と書くべきか「exit(127)」と書くべきか，詳しい理由を付して答えなさい．(10 点)

exit(127) と書くべきである．
理由：8 行は mysystem 関数を呼び出したプロセス（親プロセス）ではなく，mysystem 関数内で生成された子プロセスが実行する．mysystem 関数を呼び出していない子プロセスが return してはならないので，エラー時は mysysten 関数内で子プロセスは exit で終了させるべきである．(正常時は exec されたプログラムが終了すると子プロセスが終了する．) exit しないと mysystem 関数を呼び出したプログラムを親子プロセスが二重に実行することになる．

システムログラミングII 令和4年度 前期末試験

(2022.08.04 重村 哲至)　　IE4　　＿＿＿番　氏名＿＿＿＿＿模範解答＿＿＿＿＿

付録1：テストプログラム

```
// p0.c
#include <stdio.h>
#include <unistd.h>
char *args[]={"printenv", NULL};
char *envs[]={"X=Y", "A=B", "B=C", NULL};
int main(int argc, char *argv[]) {
  execve("/usr/bin/printenv", args, envs);
  printf("finish\n");
  return 0;
}
```

```
// p1.c
#include <stdio.h>
#include <unistd.h>
char *args[]={"printenv", "A", NULL};
char *envs[]={"X=Y", "A=B", "B=C", NULL};
int main(int argc, char *argv[]) {
  execve("/usr/bin/printenv", args, envs);
  printf("finish\n");
  return 0;
}
```

```
// p2.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
  if (fork()==0) {
    printf("child\n");
    exit(0);
  } else {
    int stat;
    wait(&stat);
    printf("parent\n");
  }
  printf("exit\n");
  return 0;
}
```

```
// p3.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
  if (fork()==0) {
    printf("child\n");
    exit(0);
    printf("exit\n");
  } else {
    int stat;
    wait(&stat);
    printf("parent\n");
  }
  return 0;
}
```

```
// p4.c
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
  if (fork()==0) {
    printf("child\n");
  } else {
    int stat;
    wait(&stat);
    printf("parent\n");
  }
  printf("exit\n");
  return 0;
}
```

```
// p5.c
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
  if (fork()==0) {
    printf("child\n");
    execlp("echo", "echo", "echo", NULL);
    printf("exit\n");
  } else {
    int stat;
    wait(&stat);
    printf("parent\n");
  }
  return 0;
}
```

システムログラミング II 令和 4 年度 前期末試験

(2022.08.04 重村 哲至)　　IE4　　＿＿＿番　氏名＿＿＿＿＿＿模範解答＿＿＿＿＿＿

付録２：myprintenv プログラム

```
// myprintenv.c
#include <stdio.h>
#include <stdlib.h>
extern char **environ;
int main(int argc, char *argv[]) {
  if (##(A)##) {          // 引数が無いなら
    for (int i=0; ##(B)##; i++) {
      printf("%s\n", environ[i]);
    }
  } else {
    char *env = ##(C)##;
    if (env==NULL) return 1;
    printf(##(D)##);
  }
  return 0;
}
// 実行例
// $ ./myprintenv USER      <-- 引数あり
// sigemura                 <-- 値だけ表示
// $ ./myprintenv           <-- 引数なし
// TERM_PROGRAM=iTerm.app   <-- 全てを表示
// TERM=xterm-256color
// ...
```

付録３：mysystem 関数

```
1  int mysystem(char *command) {
2    int pid, status;
3    if (command==NULL) return 1;
4    pid=fork();
5    if (pid<0) return -1;
6    if (pid==0) {
7      execl("/bin/sh", "sh", "-c", command, NULL);
8      ##(A)##;
9    } else {
10     int r;
11     while((r=wait(&status))!=pid) {
12       if (r<0) return -1;
13     }
14   }
15   return status;
16 }
```

付録４：ヒント

環境変数の操作

```
export name         # 環境変数の追加
export name=value   # 環境変数の追加
name=value          # 環境変数の値変更
$name               # 環境変数の参照
unset name          # 環境変数の削除
```

環境変数に関係するコマンド

```
printenv [name]
env name1=value1 name2=value2 ... command
```

C 言語の関数・変数やシステムコール

```
// 環境変数に関する関数・変数
extern char **environ;
char *getenv(char *name);
int setenv(char *name, char *val, int overwrite);
int putenv(char *string);
int unsetenv(char *name);

// プログラム実行関係
int execve(char *path,
             char *argv[], char *envp[]);
int execl(char *path, char *argv0,
            *argv1, ... , argvn, NULL);
int execlp(char *file, char *argv0,
             *argv1, ... , argvn, NULL);

// プロセス生成・終了関係
int fork(void);
void exit(int status);
pid_t wait(int *status);

// その他関数
int system(char *command);
```