オペレーティングシステムの機能を使ってみよう
第１０章 UNIX シェル

UNIX のシェルとは

- CLI（Command Line Interface）方式のコマンドインタプリタ
- macOS の Finder や Windows の Explorer は GUI 方式のシェル
- CLI 版のシェルは，sh，bash，ksh，zsh，csh，tcsh など
- UNIX シェルの持つべき機能
  1. 外部コマンド（プログラム）の起動
  2. カレントディレクトリの変更
  3. 環境変数の管理
  4. 入出力のリダイレクト，パイプ（<, >, |）
  5. ジョブの管理（jobs，fg，bg など）
  6. ファイル名の展開（ワイルドカード（*，?））
  7. 繰り返しや条件判断
  8. スクリプトの実行

簡易 UNIX シェル（myshell）

- **特徴**
  C 言語で 70 行以内で記述できる簡易シェル
- **できること**
  1. 外部コマンド（プログラム）の起動
  2. カレントディレクトリの変更
- **目的**
  myshell の構造を学び，「fork-exec 方式」，「環境変数」，「リダイレクト」等への理解を深める．
- **目標**
  「環境変数の管理機能」，「リダイレクト機能」を myshell に追加できるようになる．

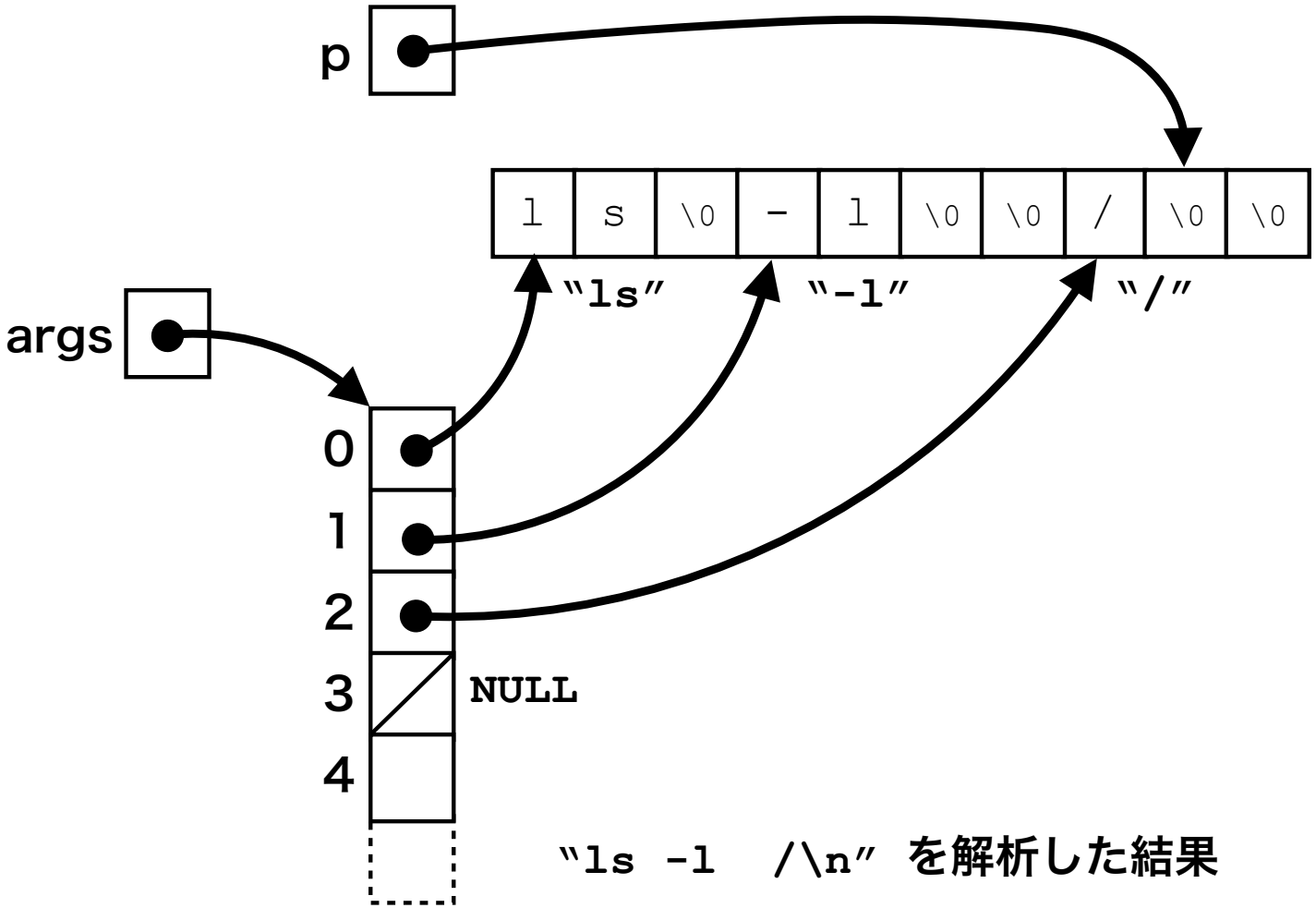
基本構造（main() 関数）

```
1  int main() {
2    char buf[MAXLINE+2];                          // コマンド行を格納する配列
3    char *args[MAXARGS+1];                        // 解析結果を格納する文字列配列
4    for (;;) {
5      printf("Command: ");                        // プロンプトを表示する
6      if (fgets(buf,MAXLINE+2,stdin)==NULL) {     // コマンド行を入力する
7        printf("\n");                             // EOF なら
8        break;                                    //   正常終了する
9      }
10     if (strchr(buf, '\n')==NULL) {              // '\n' がバッファにない場合は
11       fprintf(stderr, "行が長すぎる\n");        //   コマンド行が長すぎたので
12       return 1;                                 //   異常終了する
13     }
14     if (!parse(buf,args)) {                     // コマンド行を解析する
15       fprintf(stderr, "引数が多すぎる\n");      //   文字列が多すぎる場合は
16       continue;                                 //   ループの先頭に戻る
17     }
18     if (args[0]!=NULL) execute(args);           //   コマンドを実行する
19   }
20   return 0;
21 }
```

コマンド行の解析（parse() 関数）（1）

```
 1  int parse(char *p, char *args[]) {              // コマンド行を解析する
 2    int i=0;                                      // 解析後文字列の数
 3    for (;;) {
 4      while (isspace(*p))                         // 空白が終わるまで進む
 5        *p++ = '\0';                              //   前の文字列の終端に代用する
 6      if (*p=='\0' || i>=MAXARGS) break;          // コマンド行の終端に到達で終了
 7      args[i++] = p;                              // 文字列を文字列配列に記録
 8      while (*p!='\0' && !isspace(*p))            // 文字列の最後まで進む
 9        p++;
10    }
11    args[i] = NULL;                               // 文字列配列の終端マーク
12    return *p=='\0';                              // 解析完了なら 1 を返す
13  }
```

コマンド行の解析（parse() 関数）（2）



コマンドの実行（execute() 関数）

```
1  void execute(char *args[]) {                    // コマンドを実行する
2    if (strcmp(args[0], "cd")==0) {               // cd コマンド(内部コマンド)
3      if (args[1]==NULL)                          //   引数があるか調べて
4        fprintf(stderr,"cd の引数が不足\n");
5      else if (chdir(args[1])<0)                  //   親プロセスが chdir する
6        perror(args[1]);
7    } else {                                      // 外部コマンドなら
8      int pid, status;
9      if ((pid = fork()) < 0) {                   //   新しいプロセスを作る
10       perror("fork");
11       exit(1);
12     }
13     if (pid==0) {                               //   子プロセスなら
14       execvp(args[0], args);                    //     コマンドを実行
15       perror(args[0]);
16       exit(1);
17     }
18     while (wait(&status) != pid)                //   親は子の終了を待つ
19       ;
20   }
21 }
```

課題 No.11

1. myshell に環境変数を追加するコマンド setenv，
   削除するコマンド unsetenv を追加しなさい．
2. myshell にリダイレクト機能を追加しなさい．

```
Command: setenv A B
Command: printenv A
B
Command: unsetenv A
Command: printenv A
Command: echo aaa > a.txt
Command: cat a.txt
aaa
Command:
```