CSc 352 : Systems Programming and Unix

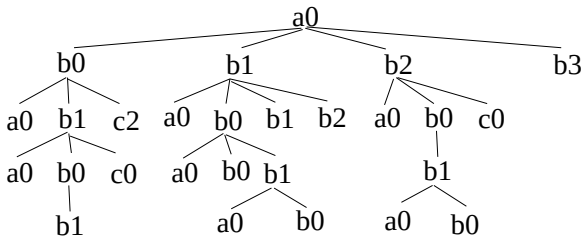
Spring 2002

Midterm Exam: Tue. Feb. 26 2002

Time : 75 mins

—SOLUTIONS—

1. Unix I [$4 \times 2.5 = 10$ points] :
The following illustrates the structure of the files within a directory `a0`:



Consider a user who starts out in the directory `a0` at the root of this tree and executes the following commands:

```
% cd b1/b0
% ls
% cd ../../b0
% ls ../b1/b0/./.././../b2/../b1/b0/b1
```

(a) What is printed out by the first `ls` command above?

Answer:
`a0 b0 b1`

(b) What is printed out by the second `ls` command above?

Answer:
a0 b0

(c) Suppose that all of the leaves in the tree shown above are "ordinary" files, i.e., not directories. How many directories are there amongst the set of files shown above?

Answer:
`10`

(d) What is the path from the root directory `a0` to where the user is after the sequence of commands shown above has been executed?

Answer:
`b0`

2. Unix II [$4 \times 2.5 = 10$ points] :
The following are the contents of a file `jabberwocky` (the line numbers on the left, outside the box, are not part of the file: they are supplied for your convenience):

```
1  'Twas brillig, and the slithy toves
2  Did gyre and gimble in the wabe:
3  All mimsy were the borogoves,
4  And the mome raths outgrabe.
5
6  "Beware the Jabberwock, my son!
7  The jaws that bite, the claws that catch!
8  Beware the Jubjub bird, and shun
9  The frumious Bandersnatch!"
```

(a) What is printed out by the command

```
wc -l jabberwocky
```

Solution: `9 jabberwocky`
(**Comment:** It's OK if the student omits the filename `jabberwocky`.)

(b) What is printed out by the command

```
head -8 < jabberwocky | tail -2
```

Solution:

```
The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
```

(**Comment:** It's OK to give just the line numbers.)

(c) What is printed out by the command

```
tail -3 jabberwocky | sort
```

Solution:

```
Beware the Jubjub bird, and shun
The frumious Bandersnatch!''
The jaws that bite, the claws that catch!
```

(**Comment:** It's OK to give just the line numbers.)

(d) What is printed out by the command

```
grep at < jabberwocky
```

Solution:

```
And the mome raths outgrabe.
The jaws that bite, the claws that catch!
The frumious Bandersnatch!''
```

(**Comment:** It's OK to give just the line numbers.)

3. C Syntax [1+2+3+4+5 = 15 points]

(a) Give declarations for:

(i) [1 point] Integer variables `x`, `y`, and `z`.

Solution: `int x, y, z;`

(ii) [2 points] An array `A` of 20 characters.

Solution: `char A[20];`

(iii) [3 points] A pointer `B` to an array of pointers to integers.

Solution: `int **B;`

(iv) [4 points] A pointer `X` to a structure containing the following fields: two integers `val` and `count`; and a pointer to a structure of the same type.

Solution:

```
struct s {
  int val, count;
  struct s *ptr;
} *X;
```

(b) [5 points] Define a macro `NewNode(x)` that behaves as follows: the argument `x` is a pointer (different uses of `NewNode()` can have arguments of different types), and `NewNode(x)` allocates data of the type pointed at by `x` and sets `x` to point to this data. If not enough memory was found, `NewNode(x)` gives an error message and exits with status -1.

Example Usage:

```
struct s { ... } *y[20];
int *A[10];
...
if ( ... ) {
  NewNode( y[2] );   /* create a new struct s and point y[2] to it. */
  NewNode( A[0] );   /* create a new int and point A[0] to it. */
}
```

Solution:

```
#define NewNode(x)  { x = malloc(sizeof(*x));\
                      if (x == NULL) {\
                         fprintf(stderr, "Out of memory\n"); exit(-1);
                      }\
                    }
```

(**Comment:** It's OK if the student: (1) omits the backslashes at the end of the lines; (2) writes `if (!x) ...` rather than `if (x == NULL) ...`; (3) uses `printf` rather than `fprintf(stderr, ...)`)

4. C Strings [$5 \times 3 = 15$ points]
Consider the following C pogram:

```
main()
{
  char A[10];
  A[0] = 'm'; A[1] = 'n'; A[2] = 'b'; A[3] = 'v';
  A[4] = 'c'; A[5] = 'x'; A[6] = 'z'; A[7] = '\0';
  myprint(A);
}
```

For each of the following definitions of the function `myprint()`, show clearly what will be printed out. If the output will be unpredictable garbage, say so.

(a) `void myprint(char *s) { printf("%s\n", s); }`

Solution: `mnbvcxz`

(b) `void myprint(char *s) { s++; s++; printf("%s\n", s); }`

Solution: `bvcxz`

(c) `void myprint(char *s) { s[4] = '\0'; printf("%s\n", s); }`

Solution: `mnbv`

(d) `void myprint(char *s) { printf("%s\n", s+8); }`

Solution: *unpredictable garbage*

(e) `void myprint(char *s) { *s++ = '\0'; printf("%s\n", s); }`

Solution: `nbvcxz`

5. Pointers [$12 + 12 = 24$ points]
Suppose that we know that the following is a legal C program, and none of the assignments in the program result in any implicit type conversions. Unfortunately, the type declarations for the variables in the program have been lost:

```
#include <stdio.h>
#include <string.h>
main()
{
  ... various declarations ...
  scanf("%s", b);
  c = strlen(b);
  x = &y;
  a = &c;
  z = &a;
  *x = z;
  **y = 24;
  printf("%d\n", c);
}
```

(a) [$6 \times 2 = 12$ points] For each of the following variables, determine its type (it's enough to write down the declaration for the variable in C syntax):

x	`int ***x`
y	`int **y`
z	`int **z`
a	`int *a`
b	`char *b`
c	`int c`

(b) [$6 \times 2 = 12$ points] The address of each variable is as indicated below. Suppose the input string given to the program is `"qwertyuiop"`. Write down the value of each of the following variables at the point just after the `printf()` in the program:

Variable	*Address*	*Value*
x	1000	`1010`
y	1010	`1100` (**Comment:** From `x = &y; ... *x = z;`)
z	1050	`1100`
a	1100	`1200`
b	1110	`"qwertyuiop"`
c	1200	`24`