CALIFORNIA STATE POLYTECHNIC UNIVERSITY
Computer Science Department
CS 2640 (3) ***T. Nguyen/F20***

PROJECT: **2**
DUE DATE: **November 13, 2020**

Description:

Write a program that implements a link list to store input lines of text. The program will then print the link list constructed.

All nodes and cstrings are to be allocated on the heap using syscall 9 (*multiple of 4*).

node structure:
address *data*
address *next*
nodes are to be added to the head of the list

The following subprograms are to be implemented:
int **strlen**(cstring *source*) : returns the length of *source* (‘\0’ not counted)
cstring **strdup**(cstring *source*) : returns a duplicate of *source* on the heap
address **addnode**(address *data*, address *next*) : returns an address to a new node initialized with *data* and *next*
traverse(address *list*, address *proc*) : traverses the *list* and calls *proc* passing the *data* of the node visit. Must use recursion to traverse from the last node to the first.
main:
prompts the user for lines of text (up to 30 characters per line)
creates a link list of these lines *llist*, the lines are to be created using *strdup*
outputs the call traverse(*llist*, *print*)
print(cstring *source*) : output *source* to the terminal

Required I/O:

```
Link List by F. Last

Enter text? Line 1
Enter text? Line 2
Enter text? Enter

Line 1
Line 2
```

Turn in:

1. Submit the source code to:
   `cp llist.s /user/tvnguyen7/cs2640-00#/BroncoName-llist.s`
   # is your section number, 1 or 2. BroncoName is the part preceding @cpp.edu in your email address.

Notes:

1. The following information is required in the beginning of every source file.

```
#
#     Name:        Last, First
#     Project:     #
#     Due:         date
#     Course:      cs-2640-0#-f20
#
#     Description:
#                  A brief description of the project.
#
```

Hints:

```
llist:   .word   0       # head of link list
inbuf:   .space  82      # up to 80 characters + \n + \0
```

main:
```
    do {
        get a line of input via syscall 8 (up to 80 characters) into inbuf
        if inbuf [0] == '\n')
            break;
        s = strdup(inbuf);
        llist = addnode(s, llist);
    } while (true);
    traverse(llist, print):
```

print(*cstring* s)
```
    output s
```

traverse(address node, address proc)
```
    while (node != 0)
        traverse(node.next, proc);
        proc(node.data);        // jalr
    }
```

int **strlen** (*cstring* s)
```
    len = 0;
    while (cs[len] != '\0')
        len++;
    return len;
```

cstring **strdup** (*cstring* s)
```
    cstring d = malloc(strlen(s) + 1); // syscall 9 - sbrk
    do {
        d[i] = s[i];
    } while (s[i] != '\0');

    return d;
```

address **addnode** (*address* data, *address* next)
```
    address node = malloc(8); // syscall 9 -  sbrk
    node.data = data;
    node.next = next;

    return node;
```

syscal 8 – read cstring (string with \0 termination)

$a0 – buffer (.space)
$a1 – length

Read from the keyboard until Enter \n and store the characters in buffer + \n + \0
If only Enter, buffer will have \n \0

syscall 9 – malloc

$a0 – number of byes
$v0 – address of the newly allocated space.

Allocate memory on the heap (dynamic memory)
$a0 needs to be multiple of 4 ((n + 3) & 0xfffffffc). For cstring, don’t forget space for \0