

## CSCB09H Worksheet: Pipe (workers & master)

In the last worksheet we wrote a program that forked one child for each command line argument. The child computed the length of the command line argument and exits with that integer as the return value. The parent sums these return codes and reports the total length of all the command line arguments together. For this worksheet, we will do the same program except the each child will communicate the length to the parent through a pipe.

```
int main(int argc, char **argv) {  
    // Declare any new variables you need
```

```
    int fd[2];
```

```
    // Write the code to loop over the command line arguments.  
    for (int i = 0; i < argc-1; i++) {  
        // Before we call fork, call pipe
```

```
        int pip;
```

```
        pip = pipe ( fd ) ,
```

```
        if (pip != -1) {  
            perror ( "pipe" );  
            exit (1);  
        }
```

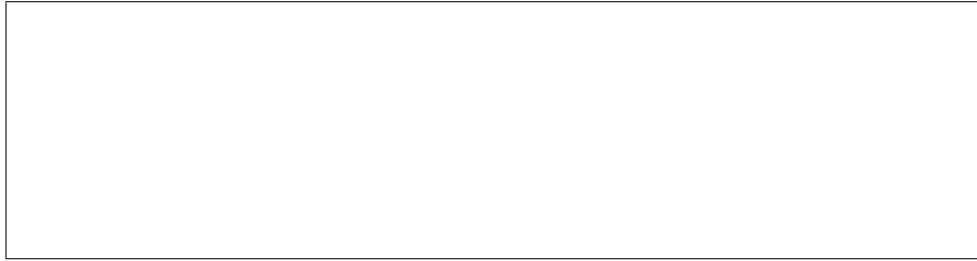
```
int result = fork();  
if (result < 0) {  
    perror("fork");  
    exit(1);  
} else if (result == 0) { // child process  
    // Child only writes to the pipe, so close reading end
```

```
    close (fd[1]);
```

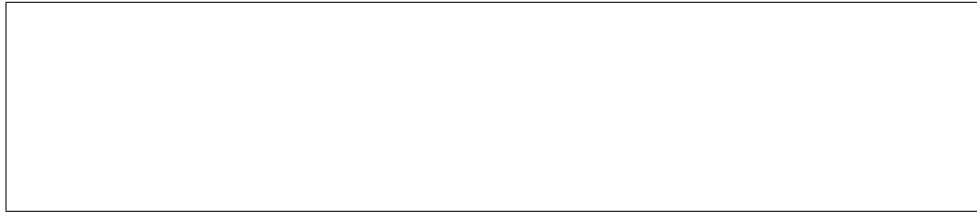
```
    // Before we forked, parent had open the reading ends to  
    // all previously forked children; so close those.
```

## CSCB09H Worksheet: Pipe (workers & master)


```
// Now do the work - write the value in binary to the pipe
int len = strlen(argv[i]);
```



```
// Close the pipe since we are done with it.
```



```
exit(0); // Don't fork children on next loop iteration
} else {
    // In the parent, but before doing the next loop iteration,
    // close the end of the pipe that we don't want open
```



```
}

// Only the parent gets here
int sum = 0;
// Read one integer from each child, print it and add to sum
```

```
printf("The length of all the args is %d\n", sum);
return 0;
```

```
}
```