

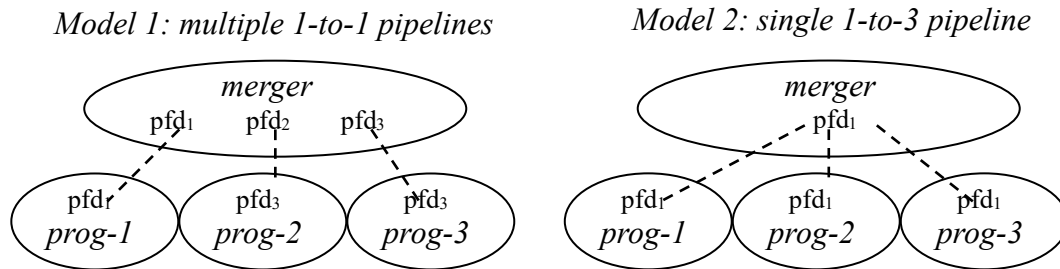
Systems Programming (Fall, 2021)

Hand-written Assignment 3 (Due on 11/10)

1. **Process control & IPC.** Alice plans to write a program *merger* that aggregates the outputs from its child processes. When issuing the following command, she wants *merger* to call *fork()* to create three child processes. Each child process calls *exec()* to execute one program, i.e., *prog-i* ($i=1..3$).

```
$ ./merger prog-1 prog-2 prog-3
```

Each child process is asked to write its output to its standard output. The output will then be sent to *merger* through a pipe. Alice takes two models into account, as shown below, where $pfid_j$ ($j=1..3$) is the file descriptor used by *pipe()*. Model 1 has three pipelines while Model 2 has only one. The two models should allow the child processes to send data simultaneously and don't generate any zombie processes. The *merger* program, i.e., *merger.c*, is not complete.



```
int main( int argc, char *argv[] )    // merger.c
{
    int i, pid;

    // Section A
    for (i=1; i<argc, i++)
    {
        // Section B
        pid = fork();
        if ( pid == 0 )
        {
            // Section C
            execlp( argv[i], argv[i], (char *) 0 );
        }
        // Section D
    }
    // Section E
}
```

Fill the following form to complete the program for supporting Model 1 and Model 2, respectively. Unused file descriptors should be closed. You could declare your own variables in Section A. Sections A~E are the places where you could put your code in *merger.c*

	Model 1	Model 2
Section A		
Section B		
Section C		
Section D		
Section E		