1. 프로세스 구조

Gcc 컴파일러를 통해 만든 실행 파일 자체는 그저 디스크에 저장되어 있는 수동적인 존재에 불과하다. 파일 형태로 존재하고 있는 프로그램이 수행되기 위해서는 리눅스 커널로부터 CPU등의 자원을 할당받을 수 있는 동적인 객체가 되어야 한다. 이 동적인 객체가 **프로세스** 이다.

결국 프로세스는 동작중인 프로그램이며, 커널로부터 할당받은 자신만의 자원을 가지고, CPU가 기계어 명령들을 실행함에 따라 끊임없이 변화하는 동적인 존재이다. 커널이 시스템에 존재하는 여러 개의 프로세스 중 CPU라는 자원을 어느 프로세스에게 할당해 줄 것인가를 결정하는 작업을 **스케쥴링** 이라고 부른다.

32bit CPU의 경우 운영체제는 각 프로세스에게 총 4GB 크기의 가상공간을 할당한다. (2^32= 4G)
리눅스는 이 중에서 0~3G 공간을 사용자 공간으로 사용하고 나머지 1G를 커널 공간으로 사용한다.

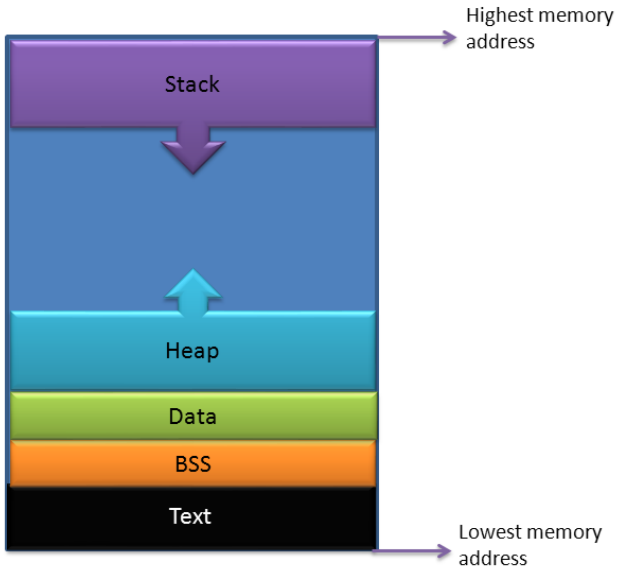

Figure : Process memory organization

폴더에 같이 있는 simple.c 를 참조하면 쉽다.

사용자 프로그램 중에서 **명령, 함수** 등으로 구성되는 텍스트 영역은 프로세서의 주소 공간 중 가장 하위 공간을 차지한다.

한편 함수의 **지역변수** 등을 담는 스택 영역은 사용자 공간과 커널 공간의 경계 위치 부터 아래 방향으로 공간을 차지한다.

또한 프로세스는 수행 중에 **동적으로 메모리 공간을 할당** 받을 수 있다. 이 때 메모리가 할당되는 공간을 힙 영역이라 부르며 힙은 데이터 영역의 다음 부분을 차지한다. (아래서 위쪽 방향으로 자라난다).

데이터 영역= 전역 변수

결국 프로세스는 크게 텍스트, 데이터, 스택, 힙 이라는 네 영역으로 구분할 수 있다. 이 때 각 영역을 **세그먼트** 또는 가상 메모리 객체(vim area struct) 라고도 부른다.

2. 프로세스 생성과 수행

프로세서는 어떻게 생성될까? 바로 fork() 또는 vfork()를 통해서이다. 폴더의 fork.c 를 확인해보자.

fork: 프로세서를 생성하고자 할 때 fork 함수를 사용하면 된다.
fork 함수를 호출하는 프로세스는 부모 프로세스가 되고 새롭게 생성되는 프로세스는 자식 프로세스
fork 함수에 의해 생성된 자식 프로세스는 부모 프로세스의 메모리를 그대로 복사하여 가지게 됩니다.

즉 프로세스가 따로 생성되면 주소 공간을 포함하여 이 프로세스를 위한 모든 자원들이 새로이 할당 됨을 알 수 있다. 따라서 자식 프로세스의 연산 결과는 자식 프로세스 주소 공간의 변수에만 영향을 줄 뿐 부모 프로세스 주소 공간의 변수에는 영향이 없으며, 결국 지역 변수, 전역 변수 등의 값이 다르게 출력된다는 것이다.

clone: 새로운 쓰레드 생성할 때 사용 (clone.c 확인)
fork와 달리 새로운 프로세스를 생성한 것이 아니다. 따라서 새로 만들어진 쓰레드는 자신을 생성한 태스크와 동일한 pid를 갖는다. 그리고 이 쓰레드는 함수의 끝을 만나면 종료된다.

1. 생성된 쓰레드와 생성한 쓰레드는 서로 같은 주소 공간을 공유한다.
2. 같은 프로세스에서 새로운 쓰레드를 생성할 경우 기존 쓰레드와 생성된 다른 쓰레드가 함께 동작한다(멀티쓰레드)
3. 자식 쓰레드에서 결함이 밥ㄹ생하면 그것은 부모 쓰레드로 전파된다.

결국 쓰레드 모델은 자원공유에 적합하며, 프로세스 모델은 결함 고립에 적합한 프로그래밍 모델이다.