The halting set K is a fundamental concept in computability theory. It is defined as follows:

$K = \{x \in \mathbb{N} \mid \varphi_x(x) \downarrow\}$

In other words, K is the set of natural numbers x for which the Turing machine with Gödel number x halts when given input x.

(Note: why the letter K was chosen for this set, it comes from the German word "Konfusion," which means confusion or undecidability. The concept of the halting set and its undecidability was first introduced by Alan Turing in his 1936 paper "On Computable Numbers, with an Application to the Entscheidungsproblem." The choice of the letter K is attributed to Stephen Kleene, who used it in his book "Introduction to Metamathematics" (1952) to denote this set.)

Theorem: K is not recursive

We will prove by contradiction that K is not recursive.

Proof:

Assume, for the sake of contradiction, that K is recursive. Then its characteristic function $\chi_K : \mathbb{N} \to \mathbb{N}$, defined as follows, is computable:

$\chi_K(x) = 1$ if $x \in K$
0 if $x \notin K$

Now, consider the following function $f : \mathbb{N} \to \mathbb{N}$:

$f(x) = 1$ if $\chi_K(x) = 0$
$\uparrow$ if $\chi_K(x) = 1$

(Here, $\uparrow$ denotes that the function is undefined, i.e., it does not halt.)

Since χ_K is assumed to be computable, f is also computable. By the s-m-n theorem, there exists a computable function $s : \mathbb{N} \to \mathbb{N}$ such that for all $x \in \mathbb{N}$,

$\varphi_{s(x)}(y) = f(x)$ for all $y \in \mathbb{N}$

Now, let $n = s(n)$ for some $n \in \mathbb{N}$. (Such an n exists by the Kleene's second recursion theorem.) Consider the behavior of $\varphi_n(n)$:

If $n \in K$, then $\chi_K(n) = 1$, so $f(n) = \uparrow$, and thus $\varphi_n(n) = \uparrow$.
But this means that $n \notin K$, which is a contradiction.

If $n \notin K$, then $\chi_K(n) = 0$, so $f(n) = 1$, and thus $\varphi_n(n) = 1$.
But this means that $n \in K$, which is again a contradiction.

Therefore, our initial assumption that K is recursive must be false. Hence, K is not recursive.